UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
BarcelonaTech
FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

Conducción automática mediante redes de imitación en un entorno virtual

Erik Miedes Bragado

Grado en Ingeniería Informática
Especialidad de computación

Trabajo de Fin de Grado
Director: Juan Climent Vilaró
Departamento: Ciencias de la Computación

2019
Defensa: 4 de julio de 2019

Resumen

La conducción autónoma es una de las ramas de la inteligencia artificial que más se está estudiando en los últimos años. Las grandes compañías automovilísticas equipan ya sus vehículos con sistemas cada vez más avanzados, y los coches totalmente autónomos están cada vez más cerca de convertirse en una realidad.
En este trabajo, se desarrolla un modelo de conducción automática mediante aprendizaje por imitación, entrenado y probado en un entorno virtual fotorrealista. El sistema inteligente se realiza mediante la mejora continuada de un modelo propuesto de forma inicial, analizando en cada iteración los fallos que se producen y proponiendo soluciones a aplicar en la siguiente fase. Una vez finalizado el desarrollo, se comprueba cómo de bien generaliza el modelo final y se detectan las situaciones en las que el sistema falla y un conductor humano debe tomar el control.
Los resultados muestran que el sistema es capaz de desenvolverse correctamente en carreteras convencionales, durante el día, tanto con tiempo despejado como con lluvia ligera. Sin embargo, tiene problemas para conducir correctamente en vías urbanas, autopistas y bajo condiciones de baja luminosidad (durante la noche y a través de túneles).

Resum

La conducció autònoma és una de les branques de la intel·ligència artificial que més s'està estudiant en els últims anys. Les grans companyies automobilístiques equipen ja els seus vehicles amb sistemes cada vegada més avançats, i els cotxes totalment autònoms estan cada vegada més a prop de convertir-se en una realitat.

En aquest treball, es desenvolupa un model de conducció automàtica mitjançant aprenentatge per imitació, entrenat i provat dins d'un entorn virtual fotorrealista. El sistema intel·ligent es realitza mitjançant la millora continuada d'un model proposat de forma inicial, analitzant en cada iteració les fallades que es produeixen i proposant solucions per aplicar en la següent fase. Una vegada finalitzat el desenvolupament, es comprova com de bé generalitza el modelo final i es detecten les situacions en les quals el sistema falla i un conductor humà ha de prendre el control.

Els resultats mostren que el sistema és capaç de funcionar correctament en carreteres convencionals, durant el dia, tant amb temps assolellat com amb pluja lleugera. Malgrat això, té problemes per conduir correctament en vies urbanes, autopistes i sota condicions de baixa lluminositat (durant la nit i a l'interior dels túnels).

Abstract

Autonomous driving is one of the most studied artificial intelligence fields in the recent years. Big car companies are already assembling vehicles featuring systems which are more advanced than ever before, and completely autonomous cars are close to becoming a reality in the roads.
In this final dessitation, an autonomous driving model is developed using learning by imitation, trained and tested inside a photorrealistic virtual environment. This intelligent system is developed by constantly improving an initially-proposed model, analysing in each iteration the errors, proposing solutions to solve them and applying these solutions in the next phase. Once the final model is finished, its generalisation capability is tested, detecting the situations that produce a system failure and require a human driver to take control.
Results show that the system is able to operate correctly in single carriageways, during daytime, both when it is sunny or there is a drizzle. Despite this, the system has trouble managing situations that include urban roads, highways or defficient lightning (nightime and tunnels).

Agradecimientos

Quiero agradecer este trabajo a todas aquellas personas que me han soportado durante su realización. A mi familia, que sin su apoyo hubiese sido imposible llegar hasta aquí. A mis amigos, que me han aguantado hablando de tecnicismos que ni yo entiendo, intentando desahogarme tanto en las noches de kebab. Ós meus galegos incondicionáis que estaban sempre aí para escoitarme cando o necesitaba e darme o apoio moral necesario para poder terminar o proxecto. Mención especial a M^3Tema, Oriol Moyano y Manuel F. Romaní, los mejores compañeros de carrera que me podían haber tocado sin duda alguna.

Agradecer también a Joan Climent Vilaró, director de este proyecto, por su gran ayuda.

A todos, gracias de corazón.

Índice

Índice de figuras

Índice de tablas

1. Contextualización

1.1. Introducción

En el campo de la visión por computador, una de las aplicaciones que más se está estudiando en los últimos años es la de la conducción automática de vehículos. La mayoría de las grandes marcas del mercado automovilístico están apostando ya por productos con distinto grado de autonomía (CBInsights, 2018). No obstante, este sigue siendo un problema abierto y de gran interés tanto para las compañías como para los organismos reguladores de la seguridad vial. Se trata, pues, de un campo muy activo y abierto a estudios.

En este trabajo, se desarrolla un sistema inteligente de conducción autónoma en un entorno virtual fotorrealista, usando para ello el aprendizaje por imitación (apartado 1.2.2). Para ello, se parte de un primer modelo y se van aplicando mejoras en base a los problemas hallados. Tras la consecución del modelo final, se comprueba cuán bien generaliza, y se detectan las situaciones en las que el sistema tiene problemas, para averiguar en qué momentos sería necesario que un conductor humano se pusiese a los mandos del vehículo.

1.2. Conceptos básicos

En esta sección, se explican todos aquellos conceptos básicos requeridos para entender el proyecto y el contexto en el que éste se enmarca.

1.2.1. Conducción autónoma

Pese a que se trata de un término cuyo uso está cada vez más extendido, debe definirse qué es exactamente la conducción autónoma y cuáles son sus categorías o niveles.

La *S*AE (*Society of Automotive Engineers*), diseñó una categorización por niveles (García Márquez, R. M.) (TechRepublic, 2016) que fue posteriormente adoptada por organismos oficiales como la DGT (Dirección General de Tráfico) o la NHTSA (National Highway Traffic Safety Administration). Según esta clasificación, existen cinco tipos de conducción autónoma, y las que siguen son sus definiciones:

- **Nivel 0:** no existe ningún tipo de automatización, todos los mandos son controlados por el conductor humano.
- **Nivel 1:** el vehículo cuenta con algún sistema de ayuda a la conducción, como pueden ser el control de crucero o la aceleración automática hasta determinada velocidad seleccionada por el conductor.
- **Nivel 2:** implica la existencia de algún sistema que permite al vehículo acelerar, frenar y girar hasta cierto grado, de forma autónoma. Entran en esta

categoría aquellos coches capaces de mantenerse en un carril mientras mantienen una velocidad específica a cierta distancia del vehículo que le precede.

- **Nivel 3:** el vehículo está preparado para reaccionar ante ciertas situaciones críticas, como cambiar de carril o frenar en caso de detectarse algún tipo de obstáculo en la vía. La atención del conductor sigue siendo necesaria por si una intervención fuese necesaria en caso de emergencia.
- **Nivel 4:** existen sistemas que permiten al vehículo conducir de forma completamente autónoma, trazando rutas, tomando decisiones en desvíos y respondiendo ante cualquier situación crítica. Pese a ello, el conductor sigue siendo necesario para tomar el control en caso de fallo del sistema.
- **Nivel 5:** el vehículo cuenta con las mismas capacidades que uno de nivel 4, pero cuenta además con sistemas de respaldo en caso de fallo del sistema principal. La figura del conductor humano desaparece.

1.2.2. Aprendizaje por imitación

Para implementar el sistema de conducción que se desarrolla en este proyecto, se utiliza la técnica de aprendizaje por imitación. Se trata de un concepto bastante intuitivo que trata de imitar la forma en la que aprenden los humanos. El aprendizaje por imitación es una técnica de entrenamiento de modelos de inteligencia artificial en el que a éstos se les proporciona, como conjunto de entrenamiento, una serie de ejemplos de cómo se comporta un experto (salida del modelo) ante ciertas situaciones (entrada del modelo). El objetivo es que el sistema actúe de forma similar a como lo hace el experto en situaciones parecidas, y que trate de generalizar lo aprendido a otras situaciones no observadas durante el entrenamiento.

El aprendizaje por imitación es un técnica especialmente útil cuando se busca diseñar un sistema que realice una determinada tarea que un humano ya es capaz de desarrollar, aunque quizá no de forma óptima. No es difícil ver que esto es aplicable al caso de la conducción autónoma.

Por supuesto, existen también algunos inconvenientes. El más importante, sin duda, es el coste muchas veces prohibitivo que puede suponer el recoger datos de un experto, y los sistemas necesarios para ello. Por ejemplo, si se buscase diseñar un piloto de aviación inteligente mediante esta técnica, la recogida de datos supondría la contratación de un piloto y una aeronave, junto con el combustible necesario, durante varias horas de vuelo. En secciones posteriores, se analiza cómo este proyecto trata de solucionar este problema económico que se plantea.

1.3. Actores

A continuación, se analiza a quién se dirige este proyecto, quién se espera que haga uso del mismo y quién o quiénes serán los posibles beneficiados.

1.3.1. Audiencia

Este trabajo es una memoria con todo el proceso de desarrollo del modelo de conducción autónoma de vehículos, que analiza los problemas hallados durante su construcción y cómo de bien generaliza la solución hallada. Así pues, con esto en mente, hay dos potenciales colectivos que podrían estar interesados en este proyecto:

- En primer lugar, las distintas empresas del sector automovilístico. Como se apuntaba en la introducción (1.1), son muchas las compañías que están apostando por sistemas inteligentes con distinto grado de autonomía o tiene proyectos de investigación en este sentido. Pueden beneficiarse de este trabajo para construir sus sistemas o, más concretamente, para anticiparse a los posibles problemas aquí documentados, antes de toparse con ellos.

- La comunidad científica, tanto estudiantes como otros investigadores, pueden usar los resultados aquí presentados para seguir desarrollando el trabajo en el punto en el que este termina, o para saber cómo atajar los problemas que se presenten si ya se han tratado aquí.

1.3.2. Beneficiarios

Además de las partes interesadas mencionadas anteriormente, los principales beneficiarios serían los usuarios finales de los vehículos con sistemas de conducción autónoma, pues supondría una mejora en su seguridad al eliminarse el riesgo producido por el factor humano (Hancock, 2018). No obstante, no serían los únicos beneficiados:

- **Empresas automovilísticas:** el incremento en la seguridad que puede aportar la conducción automática puede llevar a un aumento de las ventas de vehículos que integren este tipo de sistemas.

- **Usuarios finales:** los usuarios de los vehículos autónomos se beneficiarían dada la clara mejora en su seguridad.

- **Compañías aseguradoras:** el decremento de la siniestralidad, conllevaría una reducción en los gastos de las compañías de seguros, por lo que éstas pueden ser una de las partes más interesadas en proyectos de investigación en el campo de la conducción autónoma.

- **Organismos reguladores del tráfico:** el objetivo de las agencias de seguridad vial y regulación de la circulación es el de velar por la seguridad de los

usuarios de las vías y, por lo tanto, sus esfuerzos se centran mayormente en reducir el número de accidentes de tráfico. Es por esto que los avances en materia de conducción automática de vehículos son sin duda de interés para estos organismos.

1.4. Regulaciones e implicaciones legales

El campo de la conducción autónoma está sujeto a una regulación en constante evolución y que, en ocasiones, es incluso inexistente. Es el caso de España, donde no hay prácticamente leyes específicas para los vehículos de conducción automática.

La DGT en su *Ley sobre Tráfico, Circulación de Vehículos a Motor y Seguridad Vial* (DGT, 2015 [A]), con fecha de 31 de octubre de 2015, tiene artículos que van sin duda en contra de los coches autónomos. El Artículo 13.1, especifica que "el conductor debe estar en todo momento en condiciones de controlar su vehículo [...]". Implícitamente se está requiriendo la figura del conductor, pero no se define con claridad si un sistema de conducción autónoma puede tomar dicho rol.
No sólo esto, sino que además en su *Ley sobre responsabilidad civil y seguro en la circulación de vehículos a motor* (DGT, 2004), con fecha de 5 de noviembre de 2004, en su artículo primero dice textualmente que "el conductor de vehículos a motor es responsable, en virtud del riesgo creado por la conducción de estos, de los daños causados a las personas o en los bienes con motivo de la circulación". De nuevo, no deja claro quién sería considerado "conductor" en el caso de un vehículo autónomo.

Así pues, la legislación Española respecto a la conducción automática de vehículos es imprecisa. En principio, viendo los anteriores artículos, parecería que los vehículos autónomos están prohibidos. Sin embargo, el 16 de noviembre de 2015, la DGT estableció el marco legal para las pruebas de vehículos con sistemas de conducción autónoma mediante autorización previa (DGT, 2015 [B]). Desde entonces, este organismo otorga permisos de dos años (extensibles en periodos de la misma duración) a fabricantes, laboratorios y universidades que estén investigando en el campo.

Este proyecto, como se ha mencionado con anterioridad, se desarrolla dentro de un entorno virtual, es decir, en una simulación. Esto significa que el trabajo no está sujeto a ninguna regulación de este tipo y, por lo tanto, no es necesario pedir autorización ninguna puesto que ningún vehículo va a circular por las vías públicas.

Por supuesto, un vehículo equipado con un sistema inteligente que permita su conducción de forma automática debe cumplir las normas de tráfico y respetar al resto de usuarios de la vía. Sin embargo, como se detalla en el (apartado 3.2), uno de los objetivos de este trabajo es el de diseñar un sistema de conducción autónoma que

circule por la calzada sin salirse y, a ser posible, por su carril. Por tanto, no se busca que respete las leyes de tráfico en lo que se refiere a señalización o prioridades. Tampoco se espera que haga frente a situaciones críticas. Por este motivo, las leyes de circulación no son de importancia en este trabajo, más allá de que el vehículo debe tratar de conducir por la izquierda (ya que el entorno virtual está basado en Reino Unido).

2. Estado del arte

Seguidamente, se comentan brevemente algunos de los trabajos realizados tanto en el campo de la conducción automática de vehículos como en el del aprendizaje por imitación, y cómo sus resultados podrían ser útiles para este proyecto.

2.1. Adición de decisiones en un sistema de aprendizaje por imitación

En el *paper* de Codevilla et al (Codevilla et al, 2018), se desarrolla un sistema de conducción como el que se busca realizar en este trabajo, además de añadir un sistema para poder agregar la intencionaldiad al modelo. Un sistema de aprendizaje por refuerzo no tiene, de por sí, la capacidad de condicionarse según una entrada con la que no ha sido entrenado. En el caso de la conducción autónoma, esto se traduce en que en las intersecciones tomará una dirección cualquiera, sin posibilidad de que un agente externo decida la ruta a seguir. Básicamente, el sistema simplemente se limita a hacer aquello que hizo el experto en una situación similar, pero no se le puede indicar explícitamente una dirección concreta.

La solución propuesta en el *paper* para este problema es la adición de una entrada más al sistema, la *intención del usuario*. Durante el entrenamiento del modelo, para cada giro o intersección, al sistema se le provee de la dirección que realmente quiere tomar el conductor en forma de cuatro comandos (seguir, ir hacia adelante, girar a la derecha y girar a la izquierda). Optan por construir una red ramificada: en lugar de que el vector de entrada del modelo contenga tanto la imagen como los datos numéricos (pedales de aceleración y freno, junto con dirección) y el comando o intención del usuario, primero se alimenta la red con los dos primeros y después se elige la salida de la misma que más se ajusta a la voluntad del usuario (Figura 1).

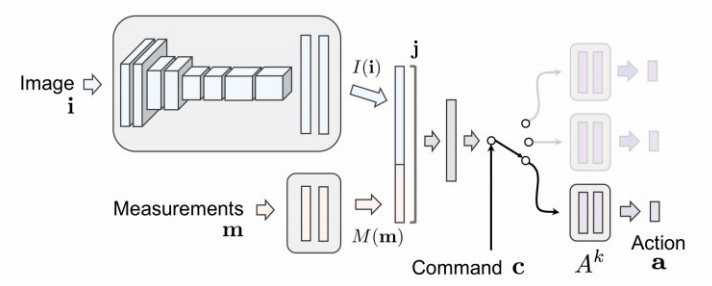


Figura 1: Esquema de la red utilizada en el *paper*. Fuente: http://vladlen.info/papers/conditional-imitation.pdf (Codevila et al, 2018)

Las conclusiones son bastante claras: se consigue el objetivo con un sistema inteligente al que se le pueden dar instrucciones para conseguir llegar a un determinado objetivo. De hecho, su sistema tiene un ratio del 0 % de giros incorrectos.

En este trabajo no se pretende desarrollar un sistema que tenga en cuenta la intención del usuario, por lo que la implementación del sistema propuesto por Codevilla et al no es relevante. Sin embargo, sí que es interesante notar que se han usado dos horas para el entrenamiento de los modelos, más complejos que los que trata este proyecto puesto que incluyen el aprendizaje de saber tomar una ruta concreta. Esto significa que con menos tiempo de conducción es posible conseguir un sistema que conduzca de forma correcta sin tener en cuenta la intención del usuario, que es lo que se pretende (apartado 3.2).

Los modelos del *paper*, además, no se desarrollan para un entorno fotorrealista con físicas precisas, como sí se hace en este trabajo. Así pues, en principio los resultados que se presentan en esta memoria son más extensibles a un entorno real.

2.2. Aprendizaje mediante imitación

Este otro trabajo de Baomar y J. Bentley (Baomar, J. Bentley, 2018), estudia la creación de un sistema inteligente para el pilotaje de aviones comerciales mediante aprendizaje automático. Aunque las aeronaves tienen piloto automático, éste sólo sirve para situarse o mantenerse a una determinada altitud o para alcanzar o seguir a una velocidad dada, y necesita al piloto para programarlos en tiempo de vuelo. El citado trabajo busca construir un sistema inteligente capaz de llevar a cabo todas las etapas de un vuelo (despegue, navegación y aterrizaje) de forma autónoma.

Para lograr su objetivo, se construyen varias redes, cada una destinada al control de distintos actuadores del aparato. Se trata de modelos muy simples, que no superan las once neuronas, y que prueban que la metodología de aprendizaje por imitación es muy eficaz en casos como este. Un sistema de programación simple con condicionales al uso permite elegir en cada momento qué red neuronal activar (figura 9 de su documento). Pese a la mayor libertad de movimiento de un avión, éste puede realizar todas sus maniobras basándose tan solo en información numérica (altura, velocidad, estado de los flaps, timón, etc). La conducción es mucho más caótica y, por el momento, no puede basarse sólo en datos numéricos y debe considerar el entorno a través de cámaras o láseres. Esto añade un plus de complejidad al problema tratado en este trabajo y hará que las redes no puedan ser tan sencillas en su topología, pero la filosofía es la misma.

En lo que respecta a los resultados, como puede verse en sus gráficas, consiguen una precisión casi total en todas las ejecuciones realizadas, imitando perfectamente a un piloto. No menciona la cantidad de datos usados para el entrenamiento, por lo que no permite estimar la cantidad de datos necesarios, como sí ha sido posible hacer con el anterior trabajo. Sin embargo, es interesante notar lo bien que funciona el aprendizaje por imitación para copiar comportamientos humanos en conducción de vehículos (en este caso aéreos).

Teniendo en cuenta la reducida cantidad de datos necesarios para el entrenamiento de un sistema de conducción automática usados en el trabajo de Codevilla et al (Codevilla et al, 2018) y que la técnica de aprendizaje por imitación funciona correctamente para controlar un vehículo complejo como un avión en el *paper* de Baomar y J. Bentley (Baomar, J. Bentley, 2018), es muy posible que un sistema inteligente como el propuesto para este trabajo sea viable, tanto técnica como económicamente.

3. Objetivos y motivación

En esta sección se describe la motivación que ha llevado a la realización de este trabajo para posteriormente detallar los objetivos concretos que se persiguen en el mismo.

3.1. Motivación

Actualmente hay una gran cantidad de estudios en el campo de la conducción autónoma de vehículos. No obstante, casi todos se centran en el desarrollo o los resultados conseguidos a partir de distintos modelos de sistemas inteligentes. Hay muy pocos informes cuyo objetivo sea analizar los problemas hallados durante el diseño de las soluciones propuestas. Por este motivo, los investigadores no tienen ninguna referencia para resolver los errores de los sistemas que diseñan ni indicaciones de cuáles son las situaciones que hacen que un modelo como el presentado en este trabajo no sepa desenvolverse correctamente.

Además, muchas de las soluciones actuales necesitan de conducción en un entorno real para recoger datos para la fase de entrenamiento de los modelos inteligentes, cosa que supone un importante gasto en el vehículo, el conductor y permisos entre otros. Así pues, en ocasiones, los sistemas inteligentes se entrenan y prueban en entornos virtuales con tal de abaratar costes y aprovechar la flexibilidad que éstos proporcionan en cuanto al control del entorno a voluntad. No obstante, cuando se opta por utilizar estos entornos virtuales, suele tratarse de simulaciones medianamente precisas en lo referente a las físicas pero muy pobres en el apartado visual (rFpro, 2018). Si los sistemas toman como entrada, única o parcial, la imagen desde el frontal del vehículo, esta imprecisión hace más difícil el poder extender el modelo a un entorno real esperando los mismos resultados que en el simulado.

Este trabajo nace de esta necesidad de documentar los problemas durante el desarrollo de un sistema inteligente de conducción autónoma para pueda ser usado como referencia para futuros trabajos, así como de la curiosidad personal por el campo de la inteligencia artificial y la conducción automática de vehículos. Además, será uno de los primeros trabajos, si no el primero, en utilizar un entorno virtual fotorrealista y con físicas de simulación para el desarrollo del sistema inteligente.

3.2. Objetivos

Los objetivos principales de este trabajo son tres:

- Desarrollar un sistema inteligente de conducción autónoma mediante aprendizaje por imitación en un entorno virtual y comprobar su rendimiento bajo distintas condiciones. El sistema final debe ser capaz de conducir sin salirse de la calzada, a ser posible por el carril de más a la izquierda (el entorno virtual está basado en Reino Unido) y a una velocidad adecuada en todo momento a las circunstancias de la vía. No es necesario que cumpla con la señalización (más allá de no ir por el sentido contrario) ni que tenga en cuenta situaciones imprevistas de riesgo como obstáculos en la calzada.
- Documentar los fallos hallados durante el proceso de desarrollo e intentar proporcionar una explicación de por qué se están produciendo. Además, se deberá proponer una solución a los mismos de forma iterativa, comprobándose si ésta es o no efectiva y explicando si lo ha sido o no y por qué.
- Detectar qué situaciones representan un peligro para el sistema, para saber así cuándo avisar al conductor humano para que éste tome el control.

4. Recursos

A continuación se detallan todos y cada uno de los recursos que han sido necesarios para la realización de este trabajo, con una pequeña descripción de su rol o utilidad dentro del mismo.

4.1. Humanos

Para llevar a cabo este proyecto, son necesarios cinco roles distintos:

- **Cabeza de proyecto:** es líder del trabajo, toma las decisiones estratégicas pertinentes y está presente en todo momento.
- **Experto en *Machine Learning*:** participa en todas las fases de implementación del modelo y su entrenamiento, así como en las pruebas del sistema.
- **Programador:** se encarga de escribir el código, bajo las directrices del expoerto en *Machine Learning*.
- **Experto (conductor):** se trata de la persona encargada de conducir el vehículo de forma adecuada para ser después imitado por el sistema inteligente.
- **Tester:** realiza las pruebas necesarias para comprobar la precisión del sistema.

4.2. Hardware

Por lo que respecta al material necesario para sacar el proyecto adelante, se necesitan los siguientes recursos:

- **Ordenador:** es el equipo, de alto rendimiento, donde se implementará, entrenará y probará el sistema inteligente. Consta de los siguientes componentes principales:
  - **Procesador:** Ryzen 7 1700, 3.7 GHz, 8 núcleos, 16 hilos.
  - **Gráfica:** Nvidia GTX 1060 6GB.
  - **Memoria RAM:** Corsair Vengance 16GB 3000MHz.
  - **Disco duro:** 2 discos de 1TB Western Digital.
  - **Placa base:** MSI B450 Tomahawk
- **Xbox One Wireless Controller:** mando inalámbrico usado para contolar el vehículo en el entorno virtual, con *joysticks* para emular el volante de forma precisa.

4.3. Software

Los recursos software utilizados en trabajo son, sin duda, los más numerosos:

- **Python:** es el lenguaje utilizado para desarrollar el sistema inteligente. Se ha optado por utilizar Python debido a su gran versatilidad, su gran número de librerías disponibles y su alta capacidad de prototipado rápido.
- **Librerías:** las librerías de terceros que se han utilizado durante el proyecto han sido las siguientes:
  - **Numpy:** utilizada para trabajar con vectores y matrices de forma más cómoda y sencilla.
  - **ImageGrab:** usada para capturar la pantalla.
  - **TFLearn:** un *wrapper* de TensorFlow que facilita su uso. Es la librería utilizada para crear, entrenar y utilizar la red neuronal.
  - **OpenCV:** utilizada para trabajar con imágenes.
  - **PyVjoy:** es la librería que permite controlar el mando virtual de vJoy.
  - **Matplotlib:** usada para crear las gráficas presentadas en esta memoria.
- **PyCharm:** el entorno de desarrollo integrado en el que se ha programado el código. Destaca su sencillez, su corrección a tiempo real, las sugerencias de código y la integración de todo el sistema de forma sencilla. También es muy útil la consola de Python proporcionada, que permite hacer pruebas in-situ antes de programar el código definitivo.
- **Forza Horizon 4:** el entorno virtual con el que se entrenará y probará el modelo inteligene. Se trata de una simulación con físicas precisas y unos gráficos fotorrealistas que imitan casi a la perfección un entorno real. Cabe destacar que está basado en Reino Unido, por lo que se conduce por la izquierda. De ahora en adelante, se referirá a este software como *FH4*.
- **Overleaf (www.overleaf.com):** le herramienta web utilizada para la creación de esta memoria con el lenguaje LaTeX.
- **LaTeX:** lenguaje usado para crear esta memoria.
- **GIMP:** usado para editar las distintas capturas del entorno virtual y la entrada del modelo presentadas en este documento.
- **vJoy:** un software que crea un mando virtual, utilizado por el sistema inteligente para controlar el vehículo.
- **XOutput2:** algunos mandos utilizan la API DirectInput, pero *FH4* sólo trabaja con XInput. Este pequeño programa, permite traducir comandos de DirectInput a XInput, de forma que el entorno virtual reconozca el mando virtual creado con vJoy.

- **Windows 10:** el sistema operativo utilizado.
- **Google Chrome:** el navegador mediante el cual se han descargado los programas necesarios y se ha accedido a las distintas referencias de este trabajo.
- **FastStone Capture:** pequeño programa utilizado para capturar regiones de la pantalla.
- **TeamGantt (www.teamgantt.com):** la herramienta web utilizada para crear los diagramas de Gantt.

5. Reconocimiento de la velocidad

La entrada al modelo consistirá en una imagen de la cámara frontal del vehículo junto con su velocidad en cada momento. Dado que *FH4* no dispone de API alguna, el valor de dicha velocidad debe extraerse directamente a partir del velocímetro situado en la parte inferior derecha de la pantalla. En este apartado, se analizara cuál es la implementación desarrollada que permite extraer la velocidad del coche a partir de los dígitos visibles en la interfaz del videojuego.

5.1. Idea general

Para averiguar cuál es la velocidad del vehículo en cada fotograma, hay que implementar un algoritmo de OCR (del inglés Optical Charachter Recognition) que reconozca los dígitos que aparecen en pantalla. Esta tarea se ve afectada por una serie de factores que facilitan su implementación. En primer lugar, los números utilizan una fuente tipográfica fija, por lo que no hay que identificar distintas variedades del mismo dígito. Además, aparecen siempre en la misma posición de la pantalla (figura 2) y tienen un color particular (en concreto, un valor RGB (255, 255, 255)), diferente al de cualquier otro elemento del juego. Esto permitirá tanto el ahorro de la búsqueda de los dígitos en la imagen como el filtrado por color para eliminar cualquier otro elemento distinto de los números que se pretenden reconocer.

5.2. Implementación

Nota: En esta sección se discutirán las principales características de la implementación utilizada. El código completo se encuentra adjuntado en la entega electrónica de este trabajo.

El OCR diseñado para reconocer la velocidad del vehículo funciona de la forma siguiente:

1. **Fase de entrenamiento:** se le pide al experto que conduzca a distintas velocidades [1] durante 10 segundos para poder capturar una imagen del segundo dígito[2]. Estas imágenes se filtran por color, dejando sólo los valores RGB iguales a (255, 255, 255) para después representarla con una matriz de enteros booleanos (1 ó 0) según si el valor es 255 ó 0 respectivamente. Cada una de estas matrices se guardan en un vector que representa el modelo aprendido por el sistema.

2. **Fase de uso:** tomando como entrada una imagen de toda la región de juego y el modelo entrenado previamente, el programa captura las regiones donde se sitúan los tres dígitos y les aplica el filtrado y la transformación usados en la

[1] 0, 100, 10, 20, 30, 40, 50, 60, 70, 80 y 90 km/h.

[2] La localización y el tamaño en píxeles correspondientes a cada uno de los dígitos se calculan previamente mediante un software externo de edición fotográfica (GIMP 2.0).

fase de entrenamiento para obtener las tres matrices correspondientes. Tras esto, cada una de ellas es comparada con las representaciones guardadas en el modelo. El número resultante de la predicción será aquel cuya matriz en el modelo sea más cercana[3] a la del dígito que se pretende reconocer. Se toman, además, ciertas precauciones; si el número reconocido en la primera posición es mayor que 3, no se tiene en cuenta porque se asume que el coche no puede circular a más de 300 km/h y se trata de un error.

5.3. Pruebas

Con tal de probar su eficacia, se ha desarrollado un pequeño programa cuyo código se encuentra adjuntado a la entrega electrónica de este trabajo. El sistema toma 50 capturas de los dígitos, separadas por un segundo cada una, y compara la velocidad reconocida por el sistema con la identificada por el usuario (velocidad real).

De las 50 muestras, todas ellas se han identificado de forma correcta. Esto significa que el sistema tiene una fiabilidad del 100 %.

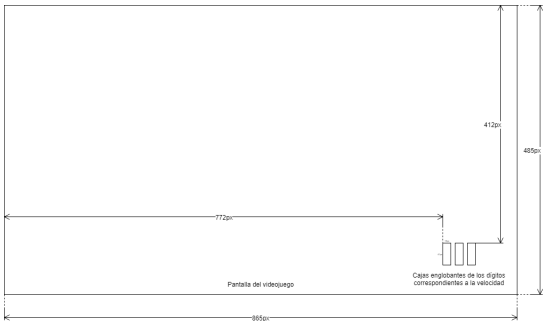


Figura 2: Diagrama de la pantalla de juego con los tres dígitos correspondientes a la velocidad y acotado. Fuente: elaboración propia.

[3]La distancia entre dos matrices es el número de posiciones distintas entre sí.

6. Diseño de experimentos

Una vez se haya diseñado un modelo suficientemente bueno, se procederá a ponerlo a prueba bajo distintas situaciones para ver cuán bien generaliza, y cómo se puede extender el sistema a distintos entornos. En esta sección, se describen los experimentos y sus condiciones:

- **Conducción por vía interurbana desconocida:** el vehículo se conducirá durante 5 minutos por una vía convencional interurbana que no haya visualizado durante el entrenamiento. El objetivo es comprobar si sabe generalizar a este tipo de vías, normalmente sinuosas y a veces estrechas, manteniéndose en su carril y sin salirse de la calzada.
- **Conducción por autopista:** se conducirán 5 minutos por una autopista. A ser posible, el vehículo debería ir por el carril de más a la izquierda (conducción británica) y debería llevar una velocidad adecuada a este tipo de vías (no menos de 50 km/h).
- **Conducción por vía urbana:** de nuevo, el experimento consiste en que el sistema conduzca durante 5 minutos dentro de poblado. Será interesante aquí ver cómo responde a las distintas situaciones características de la ciudad: carreteras más estrechas, más densidad de tráfico y edificios circundantes.
- **Conducción nocturna:** por último, se conducirán 5 minutos durante la noche, por vía interurbana. El objetivo es comprobar si el sistema es capaz de seguir la calzada a partir de la iluminación que ofrecen los faros del vehículo, y si es capaz de extrapolar esta conducción a partir de la conducción diurna.

7. Modelo inicial

A continuación, se describirá la topología y los parámetros del modelo inicial del sistema de conducción autónoma. Se trata de una red simple a partir de la cual poder ir añadiendo pequeñas mejoras para poder ver cómo éstas afectan al rendimiento del sistema y, por tanto, a la conducción automática. Se comprobará si se puede tomar el modelo como final con una pequeña prueba de 5 minutos por la ruta de la figura 14. En caso positivo, se procederá a analizar cómo de bien generaliza, y en caso contrario se investigarán los errores y se propondrán soluciones a los mismos.

7.1. Topología y características

La red neuronal de este primer modelo base toma como entrada la imagen de la cámara frontal del coche en escala de grises (un valor por píxel, entre 0 (negro) y 1(blanco)) con una resolución de 64 píxeles de ancho por 36 de alto, más la velocidad del vehículo. Por tanto, la topología queda de la siguiente forma:

- **Capa de entrada:** una neurona por cada píxel de la imagen proporcionada desde la cámara frontal del coche ($64 \cdot 36 = 2304$ neuronas), más una neurona para la velocidad.
- **Capas ocultas:** dos capas ocultas con 64 neuronas cada una, conectadas totalmente entre ellas y las capas de entrada y salida.
- **Capa de salida:** 3 neuronas, correspondientes a los valores del estado del volante, el pedal del acelerador y el del freno.

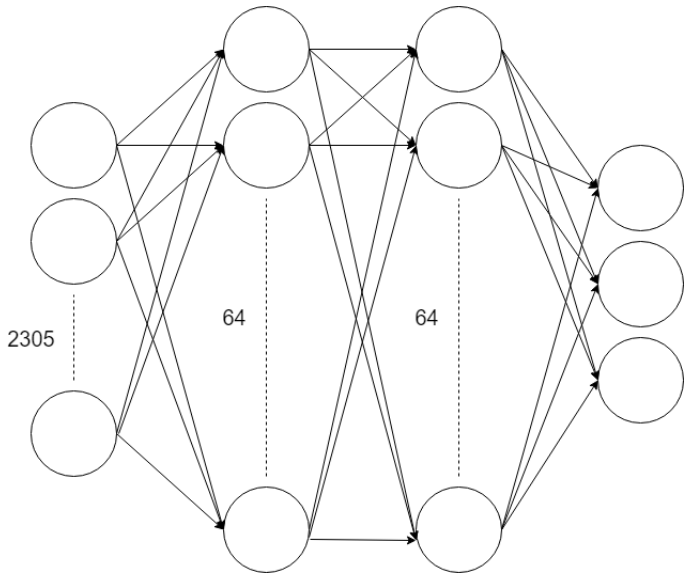


Figura 3: Esquema de la red neuronal del modelo inicial. Fuente: elaboración propia.

Los píxeles de la imagen de la cámara frontal del vehículo pueden adoptar 256 valores distintos, desde 0 para negro hasta 255 para blanco. Para estandarizar el valor de dichos píxeles, se utiliza la fórmula (1), de forma que el intervalo [0..255] pasa a ser [0..1].

$$valor\_pixel\_nuevo = \frac{valor\_pixel\_viejo}{255} \tag{1}$$

Por su parte, la velocidad también se escalará apropiadamente para que se mantenga dentro del rango [0..1]. Para ello, se supone que el vehículo no pasará de los 80km/h (durante el entrenamiento se conducirá a una velocidad máxima de 50km/h), y por lo tanto la velocidad se escala de forma sencilla según la ecuación (2).

$$valor\_velocidad\_nueva = \frac{valor\_velocidad\_vieja}{80} \tag{2}$$

La función de activación elegida para todas las neuronas de la red es la sigmoide (3), de forma que los tres valores a la salida de la misma se encuentren ya en el rango [0..1]. La técnica de optimización empleada es el *Stochastic Gradient Descent*, escogida sobre los demás métodos ofrecidos por la librería de *TFLearn* puesto que no hay un consenso claro de los expertos en la materia sobre qué método de optimización es mejor. Sería sin duda muy interesante ver cómo se comporta el mismo modelo optimizado con distintas técnicas, pero por desgracia se escapa del ámbito de este trabajo. Como función a minimizar, se ha escogido la media de errores cuadrados (4), mientras que como métrica para determinar cómo de bien explica el modelo la varianza de los datos se ha elegido el coeficiente de determinación o R^2 (X).

$$sigmoide(x) = \frac{1}{1+e^{-x}} \tag{3}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Donde y_i son los valores reales, $\hat{y}_i$ los predichos por el modelo y n el número de muestras. (4)

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{Y})}{\sum_{i=0}^{n}(y_i - \bar{Y})}$$

Donde y_i son los valores reales, $\hat{y}_i$ los predichos por el modelo, $\bar{Y}$ es la media de la variable y n es el número de muestras. (5)

7.2. Datos de entrenamiento

A continuación, se realizará una breve descripción de las condiciones en las que se han recogido los datos de entrenamiento y se hará una exploración de los mismos.

7.2.1. Descripción general

Para entrenar este primer modelo, se ha circulado durante 25 minutos siguiendo la ruta de la figura 4. El 80 % del tiempo de conducción se ha llevado a cabo en zona interurbana, mientras que el otro 10 % ha tenido lugar en poblado. Las condiciones climáticas han sido favorables, excepto por alguna llovizna ligera puntual (que no afectaba a la visión y cuyos efectos en la calzada eran nulos más allá de algunos charcos), con cielo mayormente despejado, y durante el día. Se ha intentado no superar en la medida de lo posible los 50 km/h y no se han efectuado adelantamientos por parte del conductor ni se han dado situaciones de peligro como accidentes o frenazos bruscos.

Figura 4: Ruta seguida durante el entrenamiento. Fuente: elaboración propia.

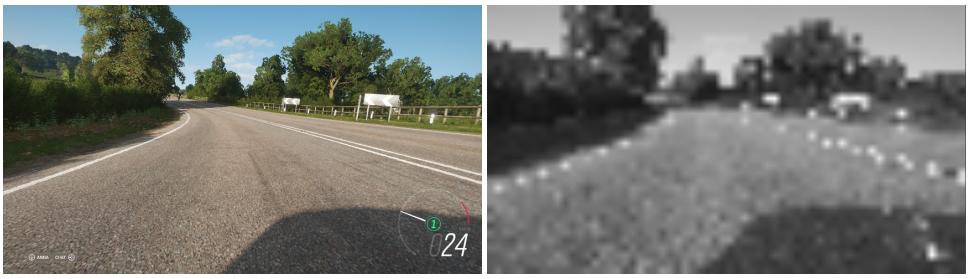
7.2.2. Análisis de los datos

Imágenes

Las imágenes obtenidas de la cámara frontal no son algo que se preste a un análisis objetivo previo que permita inferir cómo va a responder el modelo durante el entrenamiento o la predicción de nuevos valores. Sin embargo, sí es posible hacer un pequeño análisis subjetivo en base a una imagen como las usadas para alimentar la red neuronal previamente descrita.

En la figura 5, puede verse la diferencia entre la imagen real captada por la cámara del coche y la imagen de 64x36 píxeles en escala de grises proporcionada al modelo. Pese a que la resolución es muy baja, se pueden distinguir la marcas

viales y la calzada de forma bastante clara. Así pues, la red debería ser capaz de distinguir dichos elementos y predecir cómo varía el estado de los pedales y el volante cuando cambian.



(a) Imagen real (b) Imagen proporcionada al modelo

Figura 5: Diferencia entre la imagen real y la proporcionada al modelo. Fuente: elaboración propia.

Velocidad

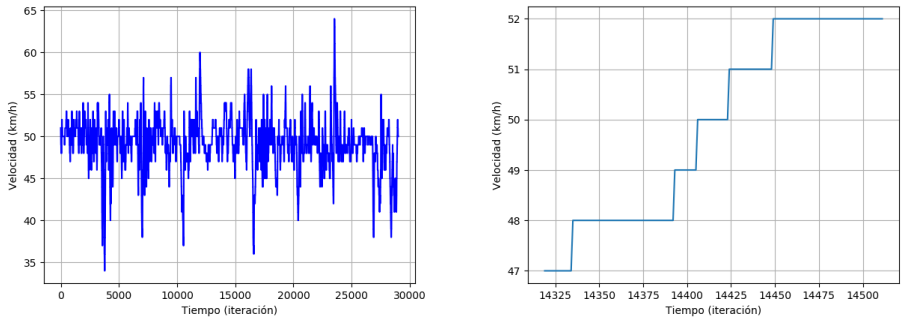
En este apartado, se analizarán los datos correspondientes a la velocidad durante el recorrido de entrenamiento, antes de estandarizar la variable.

Como se ha detallado en el apartado 7.2.1, se ha tratado de mantener una velocidad del entorno de 50km/h. La media real de velocidad durante los 25 minutos de conducción se ha situado en 49,05km/h, con una desviación estándar (7) de 3,096km/h.

Según puede verse en la figura 6 (a), hay varios outliers probablemente debidos a algún desnivel pronunciado en la calzada, a alguna incorporación en un cruce o a alguna curva muy cerrada. La velocidad máxima en el recorrido ha sido de 64km/h, mientras que la mínima se ha situado en 34km/h. Pese a que en el gráfico parece que la velocidad varía muy rápidamente, esto es sólo un efecto de representar los 25 minutos en una misma figura. Sin embargo, si se muestrean 10 segundos aleatorios, se puede observar que la velocidad varía de forma mucho más suave y paulatina (figura 6 (b)).

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \hat{X}\right)} \tag{6}$$

Donde x_i son los valores de la variable, $\hat{X}$ la media de la misma y n el número de muestras.



(a) Variación de la velocidad respecto al tiempo (20 minutos).

(b) Variación de la velocidad en 10 segundos muestreados aleatoriamente.

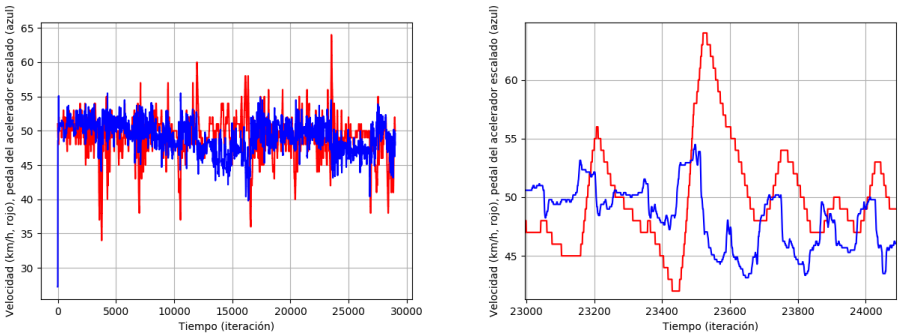
Figura 6: Variación de la velocidad respecto al tiempo. Fuente: elaboración propia.

Acelerador

A continuación se explorarán los datos correspondientes al estado del acelerador durante el entrenamiento.

Por defecto, el valor proporcionado por la API que interactúa con el mando de Xbox, se encuentra entre 0 y 65535 (pedal sin apretar y presionado a fondo, respectivamente). Para escalar y situar la variable en el intervalo [0..1] simplemente se divide por 65535. La media de esta variable se ha situado en 0,44, es decir, en el pedal del acelerador pisado en un 44 %. Por otro lado, la desviación estándar de esta variable es de 0,0442, es decir, varía alrededor de un 4,42 %. Como ocurría con la velocidad, en la figura 7 (a) se pueden observar claramente los valores extremos: hay momentos en los que el pedal se ha soltado del todo (al principio del entrenamiento, pese a que el coche está rodando, aún no se ha presionado el pedal), y en un momento concreto éste se ha pisado hasta el 56,5 % de su recorrido. Los mínimos son producidos probablemente por velocidades excesivas (por encima de los 50km/h o bien que se preveía que se iba a superar dicha velocidad), o por pendientes. El máximo ha debido ser con alta probabilidad debido a una incorporación a otra vía o por una pendiente ascendente pronunciada.

Puede comprobarse que éstas últimas hipótesis son correctas mostrando en un mismo gráfico la velocidad y el estado del pedal de aceleración (figura 7). En la figura 7 (b) puede observarse con más detalle una de las situaciones en las que el pedal del acelerador se levanta coincidiendo en el tiempo con la velocidad que llega a los 64km/h.



(a) Variación del acelerador (azul) y la velocidad (rojo) respecto al tiempo (20 minutos).

(b) Variación del acelerador (azul) y la velocidad (rojo) respecto al tiempo (zoom sobre el momento de velocidad excesiva).

Figura 7: Variación del acelerador respecto al tiempo y comparado con la velocidad. Fuente: elaboración propia.

Nota: Para escalar la aceleración por tal de poder representarla junto a la velocidad, se le ha restado su media para llevarla al cero, se ha multiplicado por 50 para que tenga una magnitud semejante a la velocidad y se le ha sumado la media de la velocidad para situarla en el mismo sitio.

Freno

En esta sección, se analizarán los datos correspondientes al pedal del freno recogidos durante el entrenamiento.

Igual que ocurría con el pedal del acelerador, esta variable también se encuentra en el rango [0..65535], por lo que se le aplica la misma transformación que a la variable del acelerador. Al analizar los datos obtenidos, destaca el hecho de que el pedal sólo se ha pisado en un momento concreto del recorrido. A juzgar por la figura 8, donde se visualiza la velocidad (rojo), el pedal del freno (azul) y el estado del volante (verde), el freno debe haberse utilizado con tal de aminorar la velocidad para tomar una curva muy cerrada a la que se llegaba con una velocidad excesiva. Esto se puede deducir porque tras el uso del freno, el volante gira y la velocidad cae rápidamente. El hecho de que no se haya precisado el uso del freno excepto en este caso puntual, se debe a que la velocidad del vehículo durante el entrenamiento ha sido tan reducida que la velocidad podía aminorarse simplemente soltando el acelerador. Así pues, esta variable es prácticamente irrelevante con el conjunto de entrenamiento proporcionado. No obstante, esto significa que el sistema no sabrá reaccionar ante situaciones que requieran de un frenado más brusco, como un coche que se cruza, un paso a nivel cerrado, o una curva extremadamente cerrada, puesto que no ha visto ninguna situación similar. Sin embargo, sabiendo que el objetivo de este proyecto es el de ver cómo se comporta un sistema según se va mejorando y analizar los posibles fallos, y dicho sistema tiene como finalidad el circular de forma correcta dentro del carril a ser posible (sin respetar señalización

vertical) (3.2), que los eventos mencionados anteriormente no se tengan en cuenta no representa un problema.

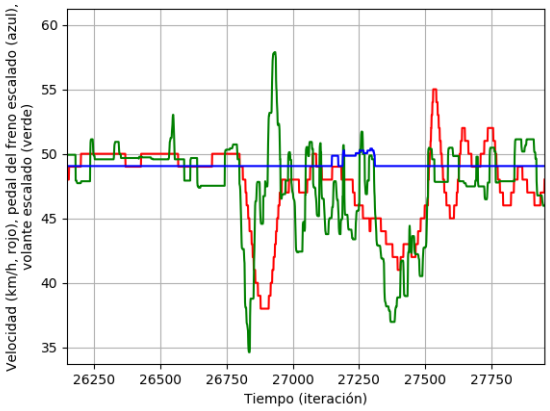


Figura 8: Datos del pedal de freno, velocidad y volante. Fuente: elaboración propia.

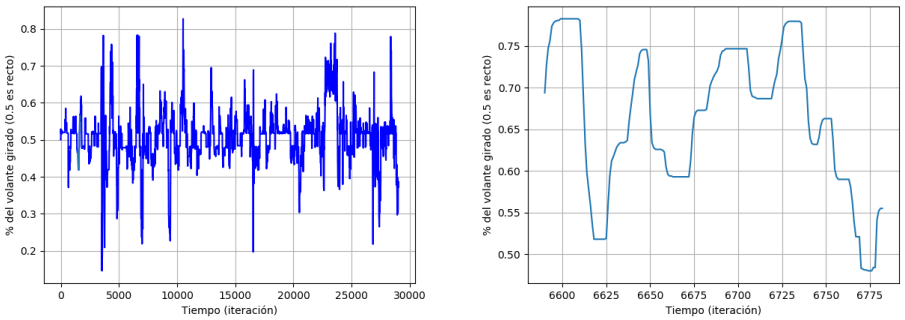
Volante

Por último, en este apartado se explorarán el estado del volante durante los 20 minutos del conjunto de entrenamiento.

De nuevo, esta variable no está limitada en el rango [0..1], si no que se encuentra en el intervalo [-32768..32767]. Así pues, se transforma de forma sencilla sumando 32768 y dividiendo luego por 65535, significando 0 el giro completo a la izquierda y 1 el giro completo a la derecha. Analizando los datos obtenidos, se observa que la posición media del volante es 0,5066 (50,66 %, es decir, el volante recto). Parece de sentido común, pues se ha circulado por un trazado sin una dirección de curva predominante (figura 4). La desviación estándar del volante es de 0,0735 (7,35 %), pero se debe tener en cuenta que una variación respecto a la media tan pequeña no sería detectada por *FH4* puesto que se encontraría en la llamda *dead zone*. Dicha zona es la que el programa considera como que el volante esté recto, aunque realmente no lo esté del todo. El valor mínimo alcanzado es de 0,146, mientras que el máximo se encuentra en 0,827, que se corresponden con los giros más pronunciados a izquierda y derecha respectivamente. Es interesante observar que el ángulo de giro más cerrado a la izquierda parece ser muy parecido al derecho, puesto que la distancia del valor mínimo y máximo a la media es similar (0,361 y 0,320 respectivamente).

En el gráfico de la figura 9, puede observarse la representación de la posición del volante respecto al tiempo. En la figura 9 (b) se muestran 10 segundos aleatorios del entrenamiento, donde se ve lo que parece ser una curva muy pronunciada a la derecha y sus pequeñas correcciones. Estas variaciones y correcciones a la hora de

girar el volante están producidas por el reducido tamaño del joystick del mando de Xbox, que hace que cualquier pequeño movimiento, voluntario o no, se refleje en los valores que adopta la variable. Esto podría confundir al modelo por la cantidad de valores que puede llegar a tomar, y una reducción en la precisión de los decimales podría beneficiar al rendimiento del sistema. Esta posibilidad se explorará más adelante (9).



(a) Variación de la posición del volante respecto al tiempo (20 minutos).

(b) Variación de la posición del volante respecto al tiempo (10 segundos aleatorios).

Figura 9: Variación del estado del volante respecto al tiempo. Fuente: elaboración propia.

7.3. Entrenamiento

En esta sección, se detallará cómo se ha realizado el entrenamiento del modelo y cuáles han sido sus resultados, antes de proceder a ponerlo a prueba durante la conducción.

En primer lugar, se han separado los datos recabados y analizados en el aparto anterior en dos conjuntos independientes: un conjunto de entrenamiento propiamente dicho, que es el que se le mostrará al modelo para que este intente ajustarse, y otro de validación, que permitirá estimar cómo de bien generaliza el modelo, antes de ponerlo a prueba con el conjunto de test. En este caso, se ha usado el 90 % de los datos recabados como conjunto de entrenamiento, mientras que con el 10 % restante se ha creado el conjunto de validación. Debido a que los datos que se recaban se van guardando en disco gradualmente (batches) pues ocuparían demasiada RAM, este 10 % son datos repartidos entre distintos momentos del recorrido de entrenamiento, y no una sección concreta como el principio o el final. Esto hace que el conjunto de validación sea más robusto y tenga en cuenta situaciones más diversas (vía urbana, interurbana, paisaje boscoso, praderas, etc).

El modelo se ha ajustado cuatro veces utilizando 5, 50, 500 y 5000 epochs por batch respectivamente, con un learning rate de 0,1. Como los pedales de aceleración y freno no suponen mucho problema por lo simple de los datos analizados en

la sección anterior, para saber cómo de bien se ha entrenado el modelo a priori, se ha comparado lo que éste predice para los datos de entrenamiento utilizados (conjunto de entrenamiento y validación) con la respuesta del conductor real. A juzgar por las figuras 10, 11, 12 y 13, se pueden sacar las siguientes conclusiones:

- **5 epochs**: se produce un claro infraajuste, con MSE de 0,0115, un RMSE (7) de 0,107 y un R^2 de 1,104. Éste último valor indica que, como se ve en la figura 10, el modelo ajusta peor que un hiperplano.
- **50 epochs:** parecen suficientes como para que el modelo se ajuste de forma más o menos adecuada. El MSE del volante se sitúa en 0,0089, dando por tanto un RMSE de 0,0944. Esto significa que el modelo tiene un error de casi un 10 % respecto al valor real del volante. El coeficiente de determinación se sitúa en 1,121, lo que podría significar que el modelo rinde peor que un hiperplano situado en la media de los datos. Esto se debe a la gran variabilidad que tienen los mismos (figura 11 (b)), que hace que el modelo no pueda ajustarse bien al "ruido" de la variable. No obstante, se puede apreciar con bastante claridad que sigue, en líneas generales, los patrones del volante. Este tipo de error podría subsanarse discretizando más la variable, eliminando el "ruido", cosa que se probará más adelante (9).
- **500 epochs:** la función ajustada sufre de forma acusada de este "ruido" (figura 12), teniendo una precisión muy pobre. El MSE se ha situado en 0,015, mientras que el RMSE ha sido de 0,121. El valor de R^2 para este entrenamiento ha sido de 2,446, significando esto que el modelo probablemente ajusta peor que el entrenado con 50 epochs.
- **5000 epochs:** los resultados son muy parecidos a los del modelo anterior, con un MSE de 0,021 y un RMSE de 0,146. Por su parte, el coeficiente de correlación se sitúa en 3,675, el peor de los cuatro modelos analizados. De hecho, hay sobreajuste en algunos puntos concretos como los de la figura 13 (b).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{7}$$

Donde y_i son los valores reales, $\hat{y}_i$ los predichos por el modelo y n el número de muestras.

Por lo tanto, se puede concluir que el mejor modelo a priori parece ser el entrenado con 50 epochs, por lo que se descarta el resto para continuar.

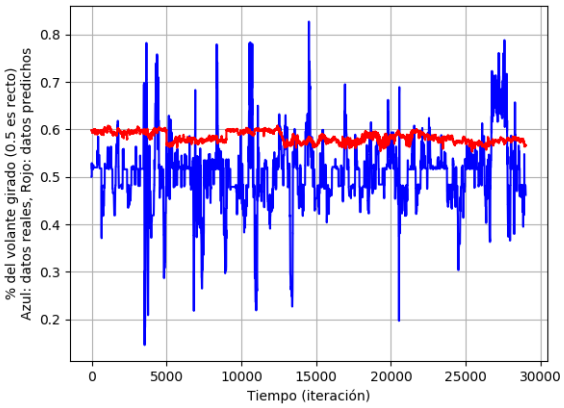
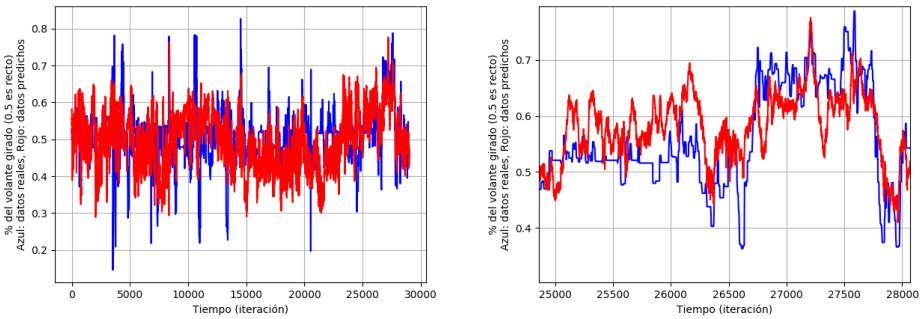


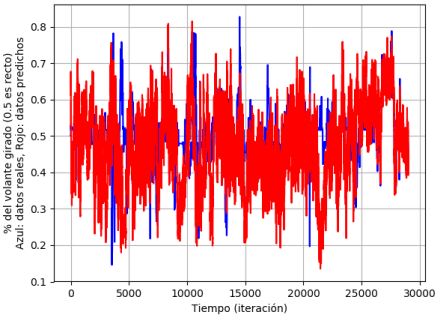
Figura 10: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 5 epochs. Fuente: elaboración propia.



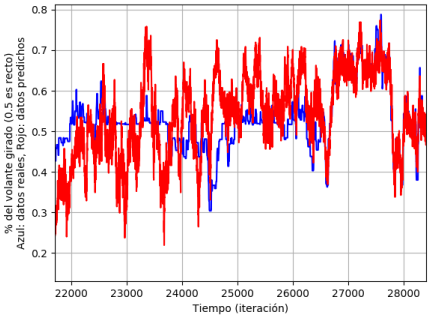
(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 11: Variación del estado del volante predicho y real respecto al tiempo, 50 epochs. Fuente: elaboración propia.

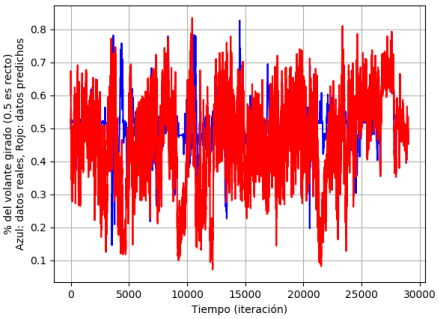


(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

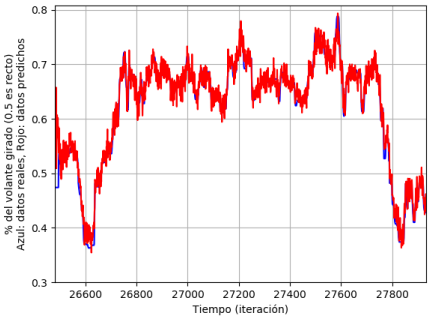


(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 12: Variación del estado del volante predicho y real respecto al tiempo, 500 epochs. Fuente: elaboración propia.



(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).



(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 13: Variación del estado del volante predicho y real respecto al tiempo, 5000 epochs. Fuente: elaboración propia.

7.4. Resultados del test

Experimento 1

En este experimento, se ha dejado al sistema conducir durante 5 minutos por parte del recorrido de entrenamiento (figura 14), con condiciones climáticas idénticas y en el mismo momento del día. Se pretende así determinar si el modelo es ya suficientemente bueno como para considerarlo modelo final. A continuación se analizarán los errores que comete el modelo y se intentará dar una explicación razonable a los mismos.

Tras el periodo de test, se ha encontrado que el sistema tiene un MTBF (8) (Mean Time Between Failures) de 16,38 segundos. Evidentemente, esto no es aceptable para un modelo de conducción autónoma adaptable al mundo real. En las figuras 15, 16 y 17 pueden verse tres ejemplos de los 15 fallos registrados. Todos ellos se han dado por salidas de la vía. El sistema no ha sido capaz de tomar de forma completa ninguna curva y ha requerido la intervención humana en todas ellas. Sin embargo, es importante notar que en muchas curvas el sistema ha mostrado la intención de girar (figuras 15 y 16), con valores de volante del entorno de 0,35 o 0,65, que si bien hacen girar al vehículo, lo hacen de forma muy sutil. En otras ocasiones, el modelo ha predicho una curva a la derecha cuando no ha habido tal (figura 17).

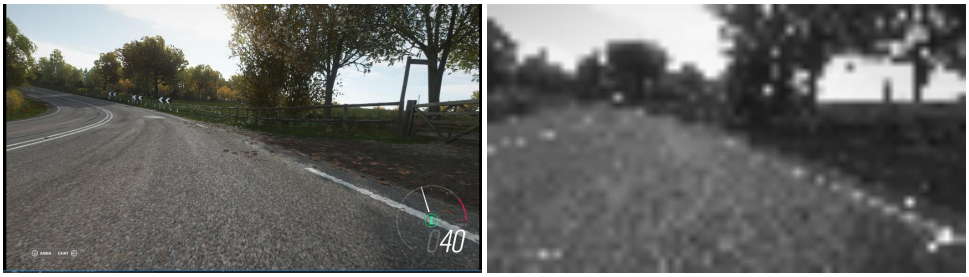
$$MTBF = \frac{\sum_{i=2}^{n} tiempo\_fallo_i - tiempo\_fallo_{i-1}}{n} \text{Siendo } n \text{ el número de fallos} \tag{8}$$

En lo que respecta a la velocidad, el vehículo ha mantenido una velocidad constante de entre 30 y 40 km/h, sin experimentar ninguna aceleración ni frenazo brusco, y llevando siempre una velocidad adecuada a la vía (que era lo que se pretendía).

Figura 14: Ruta seguida durante el entrenamiento. Fuente: elaboración propia.

(a) Imagen real. (b) Imagen vista por el sistema.

Figura 15: Curva a la derecha en la que el modelo ha mostrado intención de girar. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 16: Curva a la izquierda en la que el modelo ha mostrado intención de girar. Fuente: elaboración propia.

(a) Imagen real. (b) Imagen vista por el sistema.

Figura 17: Situación sin curva en la que el modelo ha mostrado intención de girar. Fuente: elaboración propia.

En la figura 18, se puede ver la comparación entre los datos predichos por el modelo para el recorrido de este experimento, comparados con los datos reales captados durante el entrenamiento por el mismo trazado. Se debe tener en cuenta que es imposible empezar el test en el mismo lugar exacto desde el que se inició el entrenamiento, y que la velocidad en cada uno de los recorridos ha sido distinta. Por este motivo, los dos gráficos no encajan de forma precisa, puesto que dependiendo

de la velocidad se llega antes o después a las curvas y éstas duran más o menos tiempo, cambiando también el ángulo de ataque. Aún así, se pueden identificar claramente algunas curvas, etiquetadas con números en el gráfico. Resulta evidente que el modelo ha sido capaz de identificar las curvas 1, 2, 3 y 8. Sin embargo, los giros a la derecha 5 y 6 parecen no ser detectados por el modelo. Hay además un claro fallo en el punto 9, en el que se identifica una curva a la derecha que parece no existir.

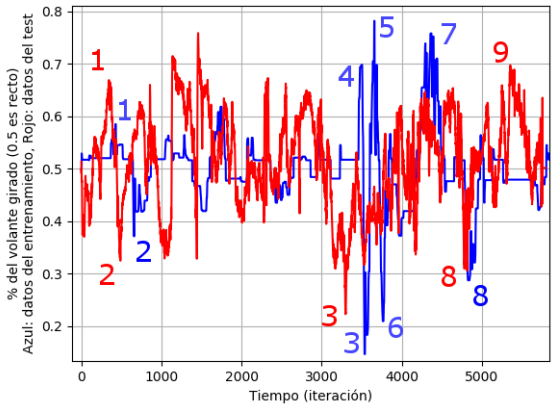


Figura 18: Variación de la posición del volante del test (rojo) y del entrenamiento (azul) respecto al tiempo. Fuente: elaboración propia.

Se puede concluir pues, que con los datos proporcionados a la red, el modelo es capaz de discernir ciertas curvas, pero se equivoca con otras muchas. No obstante, pese a que hay giros que sí identifica, no los consigue hacer debido a que el valor predicho por el modelo no es suficiente para que *FH4* lo detecte. Para comprobar la efectividad real del sistema, con giros efectivos, se aplica una función (9) en la salida de la red correspondiente al volante. Así, el porcentaje de giro se duplicará y será percibido por *FH4*.

$$volante = ((volante - 0{,}5) \cdot 2) + 0{,}5 \tag{9}$$

A continuación se detallan los resultados de la repetición del experimento una vez aplicada la función.

En esta ocasión, el MTBF ha sido de 10,12 segundos, habiéndose producido un total de 19 errores. En cuanto al tiempo entre fallos, parece un claro empeoramiento respecto al modelo anterior. Ha incrementado también el número de fallos en un 27 %. El problema principal es que el problema de las curvas inexistentes a la derecha se acentúa y se materializa, haciendo que el vehículo se salga por el borde derecho de la calzada continuamente. Esto debe producirse por un entrenamiento erróneo alentado por la falta de información suficiente como para saber cómo y

cuándo efectuar los giros. Es decir, parece ser que la información proporcionada al modelo no es suficiente como para que éste aprenda a conducir correctamente.

8. Ampliación de capas y neuronas

Una de las causas que podría estar ocasionando que la red no aprenda de forma correcta podría ser su simpleza. Por ello, en este nuevo modelo se ha decidido aumentar el número de capas y de neuronas por capa para ver cómo esto afecta a la conducción y si se aprecia alguna mejora de consideración. En esta sección se describirá la topología de la nueva red, los datos usados para entrenarla, el proceso de entrenamiento y los resultados de su puesta a prueba.

8.1. Topología y características

La entrada de esta red es igual a la de la anterior, con mismos datos y misma resolución de imagen. La topología, por su parte, queda modificada de la siguiente manera:

- **Capa de entrada:** una neurona por cada píxel de la imagen proporcionada desde la cámara frontal del coche ($64 \cdot 36 = 2304$ neuronas), más una neurona para la velocidad.
- **Capas ocultas:** cuatro capas ocultas con 512 neuronas cada una, conectadas totalmente entre ellas y las capas de entrada y salida.
- **Capa de salida:** 3 neuronas, correspondientes a los valores del estado del volante, el pedal del acelerador y el del freno.

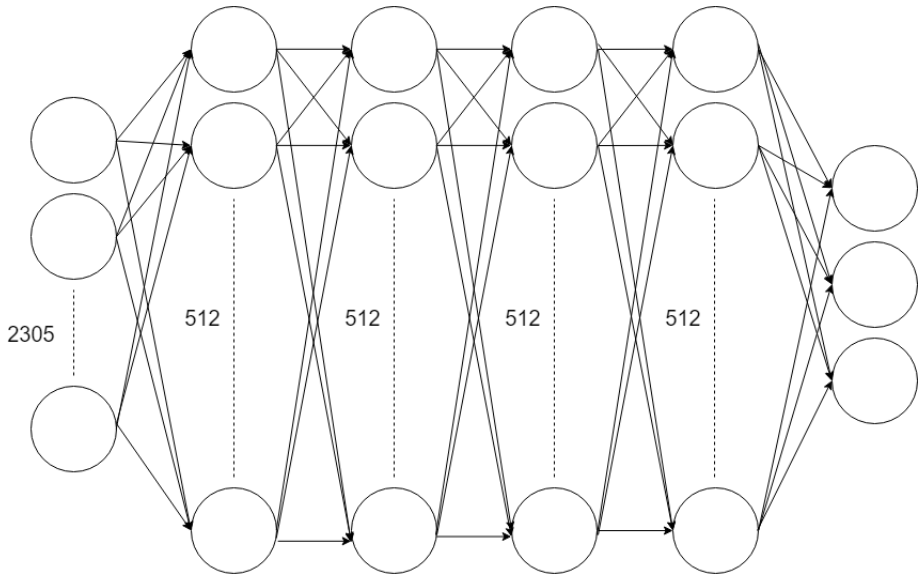


Figura 19: Esquema de la red neuronal con incremento en el número de capas y neuronas. Fuente: elaboración propia.

De nuevo, tanto los valores de los píxeles de la imagen como la velocidad son escalados usando la ecuación 1 y 2 respectivamente.

8.2. Datos de entrenamiento

Los datos de entrenamiento para este nuevo modelo son exactamente los mismos que se han utilizado para el anterior (7.2), por lo que se omite su descripción y posterior análisis y exploración.

8.3. Entrenamiento

En esta sección se detallará cómo se ha llevado a cabo el entrenamiento y se analizarán sus resultados a priori.

Igual que en modelo anterior, se han separado los datos de entrenamiento en dos conjuntos independientes, uno de entrenamiento como tal, y otro de validación para poder obtener una estimación de cuán bien generaliza el sistema. Los porcentajes restan sin variación, usando un 90 % de los datos como conjunto de entrenamiento y el 10 % restante como conjunto de validación.

El conjunto se ha ajustado cuatro veces tal y como se hizo con la red anterior: para 5, 50, 250 y 2500 epochs, con una tasa de aprendizaje de 0,1. A continuación se describen los resultados de cada uno de los entrenamientos, para permitir seleccionar un tipo de entrenamiento óptimo:

- **5 epochs:** vuelve a producirse infraajuste. El MSE es de 0,011 y el RMSE de 0,104, mientras que R^2 se sitúa en 1,022, por lo que de nuevo, el modelo ajusta peor que un hiperplano. De hecho, se produce un claro infraajuste (figura 20). La variable predicha tiene más varianza que los datos reales porque por alguna razón, tiene un sesgo hacia la derecha.

- **50 epochs:** parece que con este valor de epochs, el modelo ajusta mejor. Tiene un MSE y un RMSE muy similares a los de la red simple entrenada con el mismo valor, 0,009 y 0,0949 respectivamente. De igual forma, el coeficiente de correlación es muy parecido, con un valor de 1,1696. De nuevo, parece ajustarse en líneas muy generales a los datos proporcionados (figura 21)

- **250 epochs:** de nuevo, la función ajustada sufre de ”ruido” (figura 22), teniendo un MSE de 0,011 y un RMSE de 0,104. El R^2 es de 1,686, significando que el modelo predice peor que el entrenado con 50 epochs.

- **2500 epochs:** en este caso, el ”ruido” se acentúa aún más (figura 23). El MSE es de 0,021 y el RMSE es por tanto 0,145. Por su parte, el coeficiente de correlación se encuentra en 3,595, un valor mucho peor aún que el del modelo entrenado con 250 epochs.

Los resultados son, a priori, bastante similares a los vistos en la red más simple. No obstante, el mejor modelo vuelve a ser el entrenado con 50 epochs, por lo que se descarta el resto para continuar.

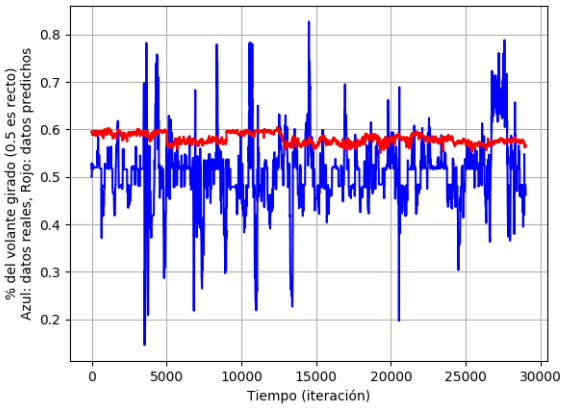
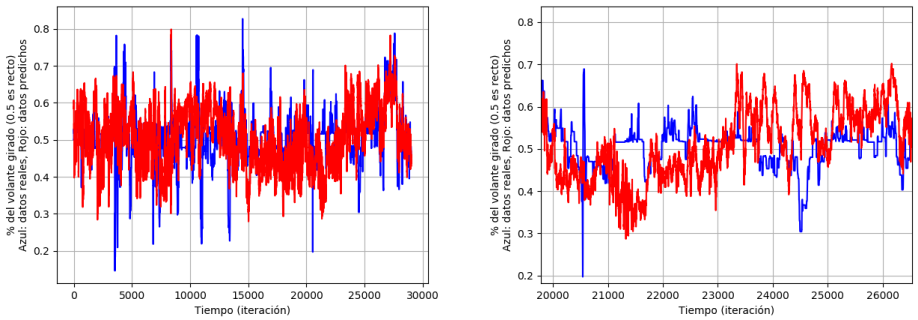


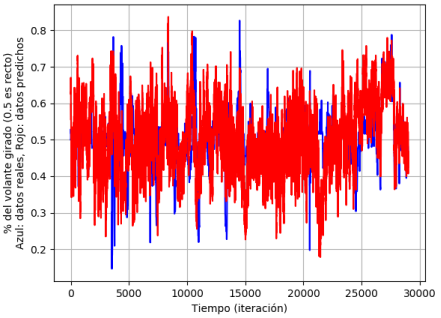
Figura 20: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 5 epochs. Fuente: elaboración propia.



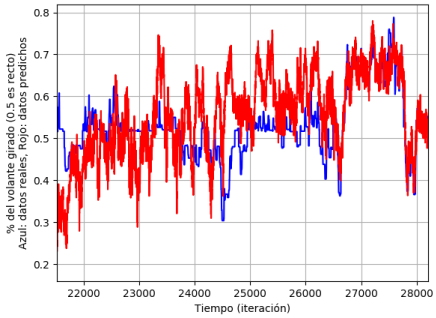
(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 21: Variación del estado del volante predicho y real respecto al tiempo, 50 epochs. Fuente: elaboración propia.

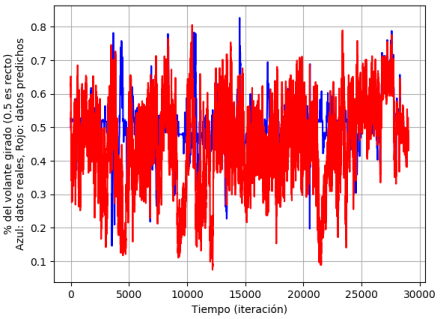


(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

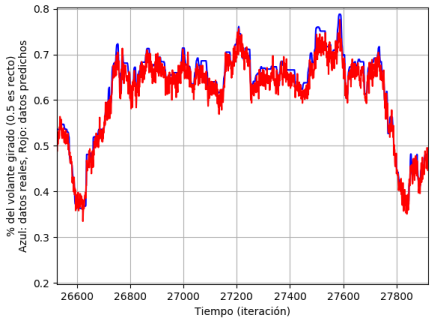


(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 22: Variación del estado del volante predicho y real respecto al tiempo, 250 epochs. Fuente: elaboración propia.



(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).



(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 23: Variación del estado del volante predicho y real respecto al tiempo, 2500 epochs. Fuente: elaboración propia.

8.4. Resultados del test

Experimento 1

Durante los 5 minutos de testeo, se han producido 14 errores. Entre ellos, han pasado una media de 17,15 segundos. Esto es un resultado mejor que el modelo simple. Sin embargo, los errores han sido muy parecidos a los vistos con el modelo previo. Sigue sin conseguir girar en la mayoría de curvas, por lo que a continuación volverá a probarse la aplicación de la función 9.

Tras aplicar el procesado a la salida predicha del volante, el MTBF ha sido de

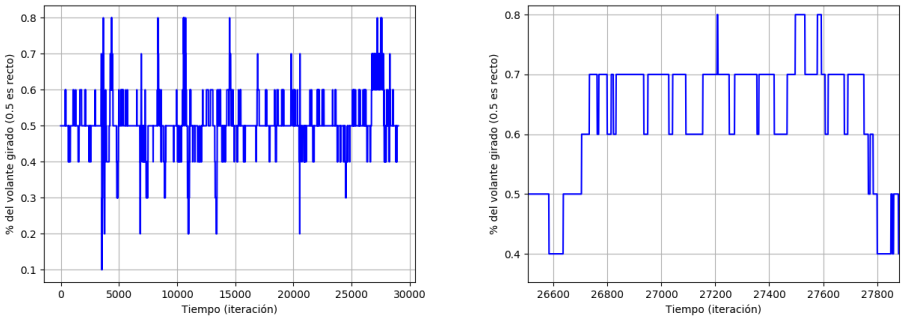
12,60 segundos, con un total de 18 errores. De nuevo, no es una mejora sustancial respecto al modelo que no usa la función, pues se observan comportamientos muy similares. La velocidad, por su parte, sigue siendo estable entre 30 km/h y 40 km/h.

Debido a que la mejora entre usar dos o cuatro capas, con menos y más neurona respectivamente, es prácticamente imperceptible en la conducción real (pese a que el MTBF o el número de errores parezcan indicar lo contrario), se ha decidido usar la topología más simple de ahora en adelante. Esto implica una mejora en el tiempo que lleva entrenar cada modelo, permitiendo una mayor exploración de posibilidades, sin importante coste aparente en el rendimiento del sistema.

9. Reducción de ruido

Una de las características que podrían estar dificultando el entrenamiento del modelo es el ”ruido” que sufre la variable del volante (figura 9), que capta incluso los pequeños movimientos que no se traducen en giros del vehículo y que son propios de la conducción humana. Para intentar reducirlo, y comprobar si esto ayuda al aprendizaje y, por tanto, al rendimiento del sistema, se discretizará la variable del volante dejándola en sólo un decimal. Así pues, la topología de la red se mantiene, así como el conjunto de datos, cuya variable del volante será transformada de la manera indicada.

En la figura 24, se puede observar cómo queda la variable correspondiente al estado del volante tras la discretización aplicada. Se puede observar que se ha conseguido reducir el número de valores que puede adoptar la variable, pero ahora la función que describen los valores de la misma es esencialmente una línea recta correspondiente a la posición del volante recto, con cambios abruptos correspondientes a curvas que duran apenas unos instantes. A continuación se entrenará de nuevo el modelo para ver si esta discretización de los datos ayuda al aprendizaje.



(a) Variación de la posición del volante respecto al tiempo (25 minutos).

(b) Variación de la posición del volante respecto al tiempo (zoom aplicado).

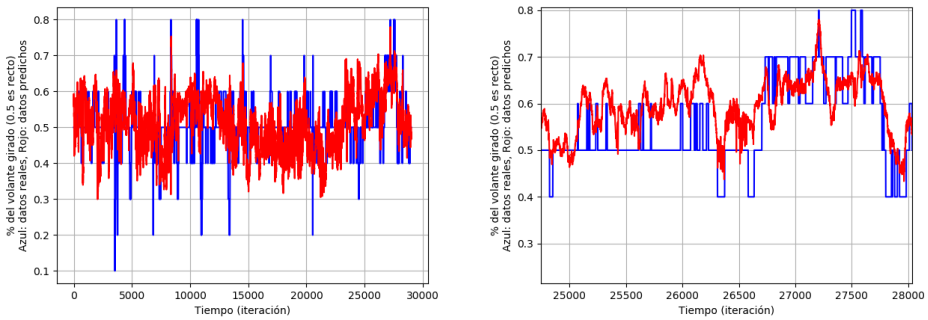
Figura 24: Variación del estado del volante suavizado respecto al tiempo. Fuente: elaboración propia.

9.1. Entrenamiento

El entrenamiento se ha llevado a cabo, como se ha mencionado con anterioridad, con el mismo conjunto de datos usados por los otros dos modelos anteriores con la transformación del volante aplicado. El learning rate sigue siendo 0,1 con objeto de ver cuál es el efecto de discretizar la dirección sin que interfieran otros factores. Respecto al número de epochs, se ha mantenido en 50 por la misma razón.

Tras el entrenamiento, se consigue un MSE de 0,009, con un RMSE de 0,097. Ambos valores son prácticamente idénticos a los obtenidos con el modelo entrena-

do con los datos sin discretizar. El R^2, por su parte, tiene un valor de $-0{,}623$. De nuevo, se trata de un valor muy parecido al del modelo anterior. En la figura 25, se puede observar que el modelo ajusta obteniendo una función casi idéntica a la observada en el anterior modelo (figura 21).



(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 25: Variación del estado del volante predicho y real respecto al tiempo, volante discretizado. Fuente: elaboración propia.

9.2. Resultados del test

Experimento 1

Tras los 5 minutos de test, se han producido 16 errores, con una media de 16,12 segundos entre ellos, con 15 errores en total. Estos valores son prácticamente idénticos a los obtenidos con el modelo simple, y el comportamiento aparente ha sido difícilmente distinguible de éste. Todo esto indica que la disminución de ruido no implica ninguna mejora ni supone ayuda alguna al entrenamiento de la red. De igual forma que en el primer modelo estudiado, el vehículo trata en ocasiones de girar pero no lo consigue debido al problema ya comentado de la sensibildiad. No obstante, no se volverá a probar la aplicación de la función ya que se ha observado el mismo comportamiento que el modelo simple sin ella. Esto significa que, de usarse la función, llevaría a unos resultados muy similares si no idénticos a los ya vistos.

10. Adición de color

Puesto que una de las características que podría ayudar a la red a conducir de forma correcta es la distinción de la carretera a partir de su color, se ha decidido añadir el color al modelo. A continuación se describirá la topología de la red, se explorará el conjunto de datos de entrenamiento, y se procederá a la puesta a prueba del modelo.

10.1. Topología y características

Esta nueva red neuronal vuelve a tomar como entrada la imagen de la cámara frontal del vehículo y su velocidad. La resolución sigue siendo de 64x36 píxeles, pues el cambio reside en lo que se muestra a la red neuronal por cada píxel. En lugar de usar un único valor correspondiente al nivel de gris, se usa el color. Para ello, se discretizan los posibles colores de la forma en la que se detallará más adelante, y se le proporciona la etiqueta del color de cada uno de los píxeles codificada en unario. Así pues, se necesitan tantas neuronas por píxel como número de colores se utilicen. La topología de la red queda, por tanto, como sigue:

- **Capa de entrada:** tantas neuronas por píxel como número de colores en los que se discretiza (en este caso 8, más una neurona para la velocidad. En total, hay $64 \cdot 36 \cdot 8 + 1 = 18433$ neuronas de entrada.
- **Capas ocultas:** dos capas ocultas con 64 neuronas cada una, conectadas totalmente entre ellas y las capas de entrada y salida.
- **Capa de salida:** 3 neuronas, correspondientes a los valores del estado del volante, el pedal del acelerador y el del freno.

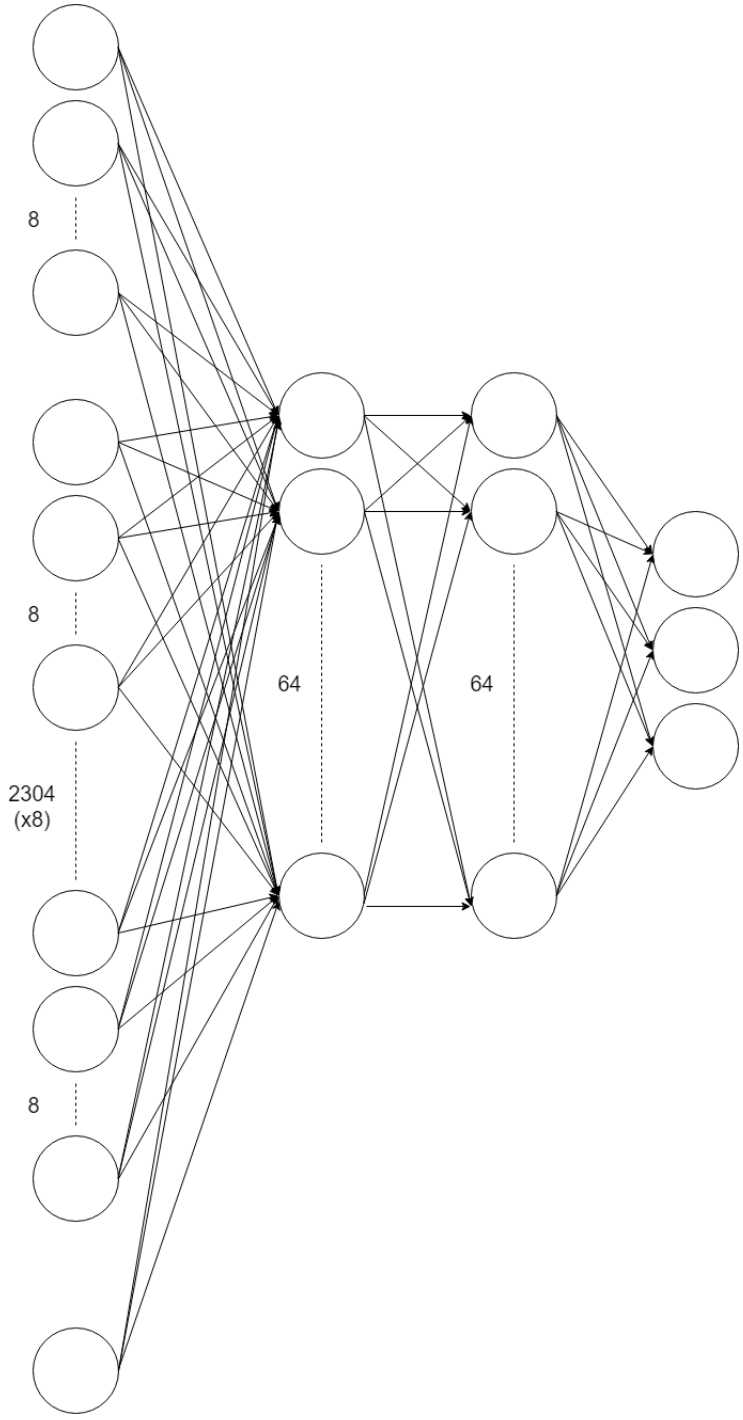


Figura 26: Esquema de la red neuronal con input de color. Fuente: elaboración propia.

Para discretizar los colores, se sigue el siguiente procedimiento:

1. En primer lugar, se cambia el espacio de colores de BGR (del inglés Blue,

Green, Red) proporcionado por la librería PIL al capturar la pantalla, a HSV (del inglés Hue, Saturation Value).

2. Tras esto, se clasifica cada uno de los píxeles en uno de los siguientes colores:
   - Negro: si el valor está por debajo del 24,7 %.
   - Blanco: si la saturación está por debajo del 24,7 %.
   - Rojo: si la tonalidad (hue) es mayor o igual que 337,5º y menor de 22,5º.
   - Amarillo: si la tonalidad es mayor o igual que 22,5º y menor de 67,5º.
   - Verde: si la tonalidad es mayor o igual que 67,5º y menor de 157,5º.
   - Azul claro: si la tonalidad es mayor o igual que 157,5 y menor de 205,5º.
   - Azul oscuro: si la tonalidad es mayor o igual que 205,5º y menor de 270º.
   - Violeta: si la tonaldiad es mayor o igual que 270º y menor de 337,5º.
3. Se representa cada píxel de la siguiente forma:

$$\begin{gathered} pixel = \overbrace{(0,0,0,...,0)}^{c} \\ pixel_{cc} = 1 \end{gathered} \tag{10}$$

Donde *pixel* es la representación del píxel, *c* es el número de colores en los que se discretiza, y *cc* es el código correspondiente al color del píxel (entre 0 y *c*)

10.2. Datos de entrenamiento

En esta sección, se describirán las condiciones bajo las cuales se han recogido los datos usados para crear los conjuntos de validación y test, y se realizará una exploración de los mismos.

10.3. Descripción general

Se ha circulado durante 25 minutos por la misma ruta seguida durante la recogida de datos anterior (figura 4). De nuevo también, un 80 % del tiempo se ha circulado por vías interurbanas, mientras que durante el otro 20 % se ha conducido el coche por ciudad. Las condiciones climáticas han sido favorables, con cielo variable y alguna llovizna puntual y anecdótica que no ha disminuido la visión ni ha afectado al comportamiento del vehículo. Que haya observado distintas condiciones climatológicas podría ayudar a que el modelo sepa cómo adaptarse a las mismas de forma adecuada. Por su parte, se ha tratado de mantener una velocidad medianamente estable de 50 km/h, adaptando siempre la velocidad a las circunstancias de la vía, sin acelerones ni frenazos súbitos. No han ocurrido situaciones de peligro durante los 25 minutos de recorrido.

10.4. Análisis de los datos

Puesto que la recogida de datos se ha llevado a cabo en la misma ruta y bajo las mismas condiciones y premisas que la anterior colecta de datos, el análisis no profundizará en aquello que sea similar, si no igual, a lo visto en el apartado 7.2.

Imágenes

La principal diferencia de este conjunto de datos respecto al anterior son las imágenes, a que se les ha añadido el color. En las figuras 27 y 28 pueden verse dos ejemplos comparativos de lo que ve la cámara y lo que se le muestra a la red neuronal. Se puede apreciar a simple vista que ahora es mucho más sencillo distinguir la calzada de aquello que no lo es. Añadiendo el color, la red sólo deberá comprender que la carretera es lo que tiene la etiqueta de ”blanco”, y que debe mantenerse a una cierta distancia de los bordes, que suelen tener etiqueta de ”amarillo” o ”verde”. En principio, esto debería facilitar el aprendizaje para la red neuronal.

(a) Imagen real (b) Imagen proporcionada al modelo

Figura 27: Diferencia entre la imagen real y la proporcionada al modelo, con color. Fuente: elaboración propia.

(a) Imagen real (b) Imagen proporcionada al modelo

Figura 28: Diferencia entre la imagen real y la proporcionada al modelo, con color. Fuente: elaboración propia.

Velocidad

De nuevo, la media de la velocidad mantenida durante el recorrido es de aproximadamente 50 km/h, concretamente de 48,115 km/h, mientras que su desviación típica se ha situado en 5,201 km/h. La velocidad máxima y la mínima han sido de 59 km/h y 16 km/h respectivamente.

Acelerador

El pedal del acelerador se ha presionado de media en un 0,457 %, mientras que su desviación estándar se ha situado en 0,047 %. El valor máximo observado ha sido de un 0,667 % mientras que el mínimo se ha situado en 0,243 %.

Freno

En este caso, el pedal del freno sí que se ha utilizado sensiblemente más que en el anterior conjunto de datos. Observando la figura 29, es interesante notar que todo su uso se ha dado en la parte final del recorrido, que se corresponde con la ciudad. Esto se debe a que, por razones de tráfico, se ha necesitado aminorar la velocidad del vehículo. La media, no obstante, sigue siendo casi nula, con un valor de 0,055 %. La desviación, por su parte, se ha situado en 0,091 %. Lo mínimo que el pedal ha sido presionado es un 0 %, y su recorrido máximo se ha establecido en 0,353 %.

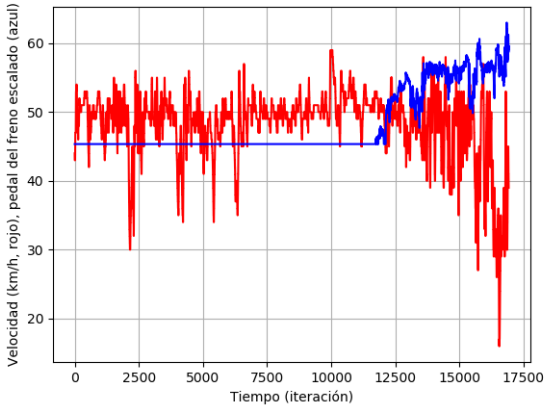


Figura 29: Cambio de la velocidad (rojo) y el freno (azul) respecto al tiempo. Fuente: elaboración propia.

Volante

La posición media del volante, como era de esperar, se ha situado en 0,503, que es la posición neutra. La desviación típica se ha situado en 0,069 %. La curva más cerrada a la derecha ha hecho que sea necesario girar el volante hasta el

0,811 %, mientras que la curva más cerrada a la izquierda ha necesitado de un giro del 0,144 %. Como ya se vio en la primera exploración de los datos, en el apartado 7.2, no hay sesgo de giros a un lado u otro.

Nota: no se han añadido las gráficas correspondientes a la evolución de las variables respecto al tiempo pues son muy similares a las analizadas en el apartado 7.2.

10.5. Entrenamiento

El entrenamiento se ha llevado a cabo utilizando un 90 % de los datos recogidos como conjunto de test, y usando el otro 10 % como datos de validación. Dichos porcentajes están repartidos de la forma previamente explicada en el apartado 7.3. El learning rate, por su parte, se ha mantenido fijo en 0,1. De igual forma que se hizo con las dos primeras iteraciones de modelo estudiadas, se ha decidido probar el entrenamiento de este nuevo modelo con cuatro epochs distintas, y elegir aquella cuyos indicadores apuntan como más favorables:

- **5 epochs:** se produce un claro infraajuste, como se puede ver en la figura 30. El valor de R^2 es de 0,097, por lo que, como es de esperar, con este entrenamiento el modelo no explica la variable en absouluto.
- **50 epochs:** a diferencia de lo que ocurría con el modelo simple entrenado con 50 epochs, ahora parece que, según se observa en la figura 31, el modelo es capaz de ajustar mucho mejor. El MSE se sitúa en 0,004, dando un RMSE de 0,065, los valores más bajos hasta ahora. El coeficiente de determinación tiene un valor de 0,704, es decir, un 70,4 % de la varianza de los datos está explicada por el modelo. Este valor es sin duda remarcable, pues mejora por mucho cualquier modelo visto hasta el momento.
- **250 epochs:** en este caso, el MSE es de 0,006, con un RMSE de 0,078. El R^2, de 1,291, podría indicar que el modelo predice peor que un hiperplano. No obstante, observando la figura 32, podemos concluir que, a priori, no es así. La función parece que ha ajustado razonablemente bien.
- **2500 epochs:** se produce un claro sobreajuste, pero además con un sesgo a la izquierda excepto en la parte final (figura 33). Esto debe estar producido por el intento de ajustar ese final, produciendo un MSE total de 0,015 y un RMSE de 0,121. Por último, se obtiene un R^2 de 2,737, con el que se puede concluir que no es el mejor modelo de los cuatro.

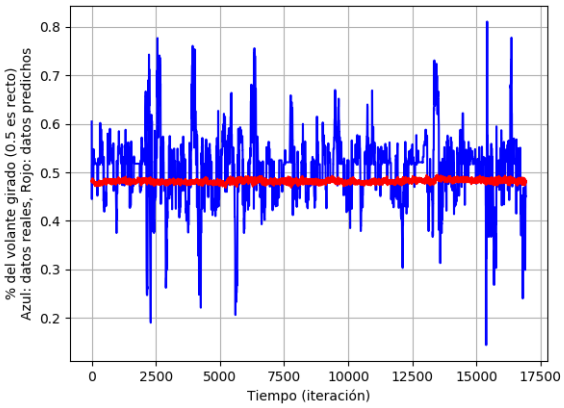
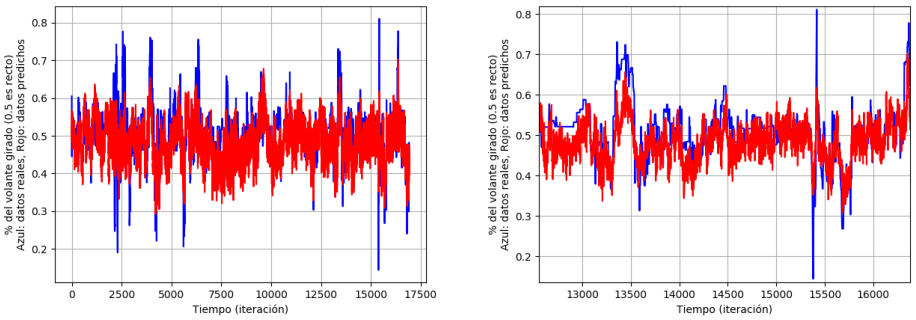


Figura 30: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 5 epochs. Fuente: elaboración propia.



(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

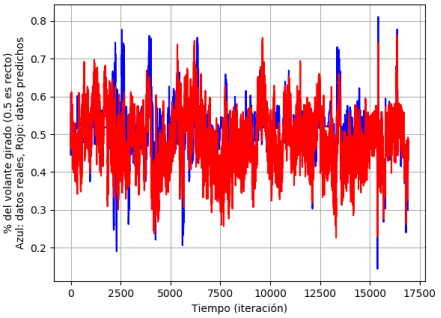
Figura 31: Variación del estado del volante predicho y real respecto al tiempo, 50 epochs. Fuente: elaboración propia.

Parece evidente que los mejores modelos son los entrenados con 50 y 250 epochs. Para tratar de discernir cuál de ellos es mejor, se realizará a continuación el experimento 1 con cada uno de ellos, y se analizarán los resultados.

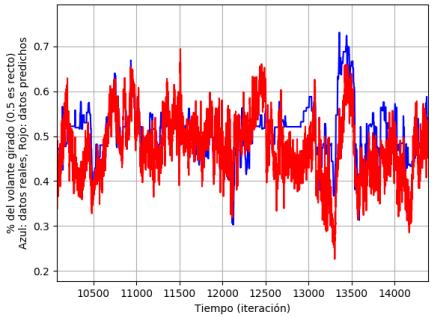
10.6. Resultados del test

Experimento 1

- **50 epochs:** el MTBF ha sido de 11,097 segundos entre fallos, con un total de 17 errores. No parece mejorar los anteriores modelos, aunque sí se presume mejor identificación en algunas curvas.

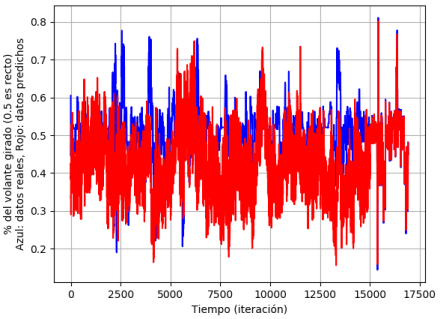


(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

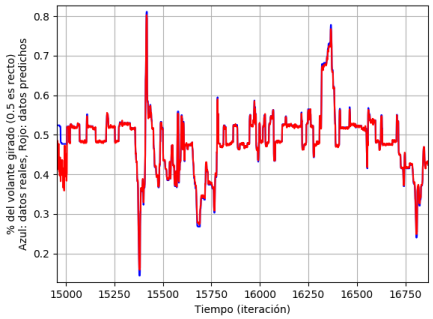


(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 32: Variación del estado del volante predicho y real respecto al tiempo, 250 epochs. Fuente: elaboración propia.



(a) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (25 minutos).

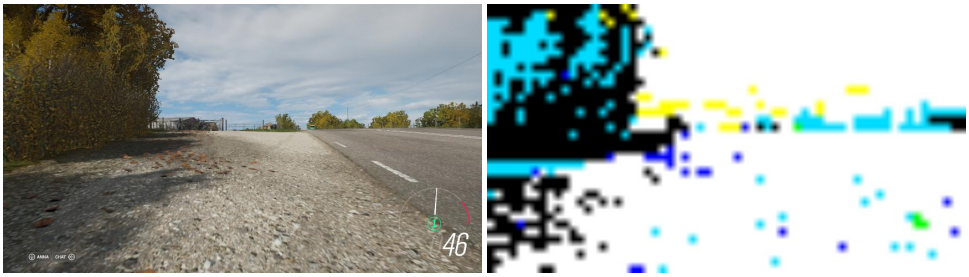


(b) Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo (zoom aplicado).

Figura 33: Variación del estado del volante predicho y real respecto al tiempo, 2500 epochs. Fuente: elaboración propia.

- **250 epochs:** el MTBF se ha situado en 11,062 segundos, con un total de 18 fallos. No obstante, se trata sin duda del mejor modelo hasta el momento. El MTBF es bajo debido a que se han producido varios fallos seguidos en una misma curva cerrada, pero el modelo ha sabido tomar casi todas las curvas que se ha ido encontrando. Sin embargo, parece haber perdido la capacidad de mantener una velocidad adecuada, acelerando fácilmente hasta los 70 km/h. Los fallos que han habido se han debido a velocidad excesiva o a partes de la vía que, como la calzada, son blancas y, por tanto, detectadas por el modelo como parte de la misma (figura 34).

Aunque podría seleccionarse este último modelo, entrenado con 250 epochs,



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 34: Situación en la que el modelo ha confundido un camino con la calzada. Fuente: elaboración propia.

como la iteración final, cabe destacar un hallazgo importante durante las pruebas de implementación que se han llevado a cabo con el fin de depurar posibles bugs de código. Para dichas pruebas, se ha comprobado que todo funciona correctamente creando un modelo entrenado con 5 minutos de datos en vía interurbana, en lugar de los 25 de urbana e interurbana que se presentan finalmente. El hecho destacable radica en que ha dado la sensación de que el modelo entrenado con menos datos ha funcionado mucho mejor que los probados posteriormente con 25 minutos como conjunto de entrenamiento. A continuación, se procederá a probar los modelos entrenados con un conjunto de datos más reducido.

11. Reducción de los datos

Por tal de comprobar la hipótesis planteada en el apartado anterior, se entrena el modelo, de idéntica topología, con 5 minutos de datos. No se hará un análisis en profundidad puesto que se trata de la primera parte del mismo recorrido con el que se entrenaron los otros modelos (figura 14), y la recogida de datos se ha llevado a cabo de la misma forma y bajo las mismas condiciones que la colecta de los 25 minutos. No tiene pues sentido un análisis de los datos, ya que se observarán las mismas características que se han visto hasta ahora. La única modificación realizada ha sido una salida de vía a la izquierda y a la derecha con posterior recuperación, por si esto sirviese a la red para aprender cómo gestionar las situaciones en las que se sale de la calzada.

Se ha entrenado con 50, 500 y 5000 epochs, y a continuación se presentan los resultados a priori. Se ha descartado el entrenamiento con 5 epochs ya que en todos los modelos se ha observado que produce un claro infraajuste.

- **50 epochs:** se obtiene un MSE de 0,002 y un RMSE de 0,049. El coeficiente de correlación se sitúa en el 0,013. Es decir, sólo un 1,3 % de la varianza de los datos está explicada por el modelo. Se trata, sin duda, de un infraajuste claro (figura 35).
- **500 epochs**: tras el entrenamiento, se produce un MSE de 0,0004 y un RMSE de 0,020. Este error mínimo, junto con un R^2 de 0,777, parecen indicar que el modelo ajusta muy bien los datos proporcionados. Esto se puede comprobar también de forma visual con la figura 36.
- **5000 epochs:** el MSE obtenido tras el entrenamiento es de tan sólo $5{,}051 \cdot 10^{-6}$, mientras que el RMSE es de 0,002. Por su parte, el coeficiente de correlación se sitúa en 0,947, su valor más alto, por debajo de 1, hasta ahora. Todo esto podría hacer pensar que es el mejor modelo, pero si se observa la figura 37, no es complicado ver que se está produciendo un claro sobreajuste.

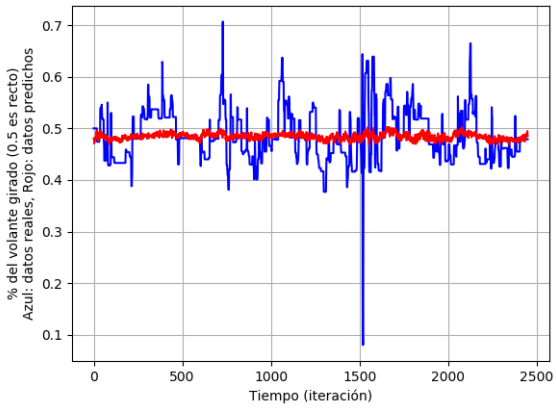


Figura 35: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 5 epochs. Fuente: elaboración propia.

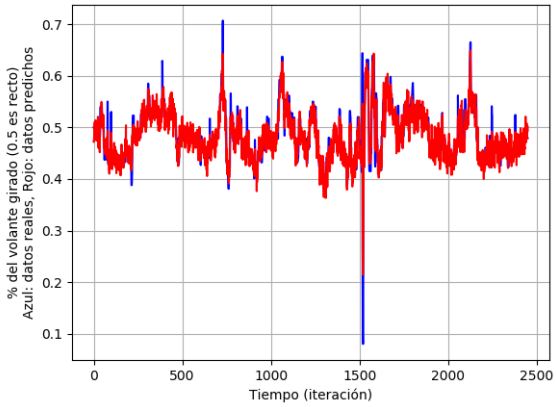


Figura 36: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 500 epochs. Fuente: elaboración propia.

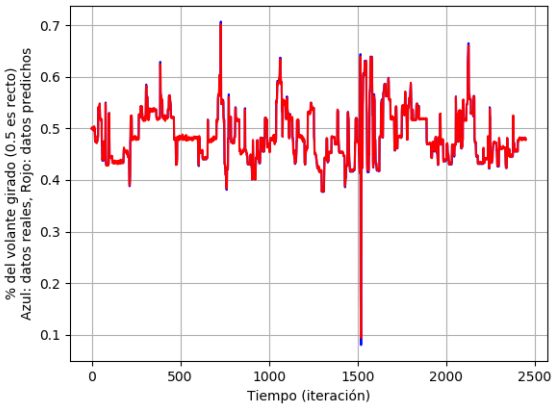


Figura 37: Variación de la posición del volante predicha (rojo) y real (azul) respecto al tiempo, 5000 epochs. Fuente: elaboración propia.

Así pues, parece que el mejor modelo sería el entrenado con 500 epochs. No obstante, también se comprobará el rendimiento con 5000 epochs, puesto que puede que, habiendo sobreajustado tanto, haya aprendido a generalizar y no sólo haya ”memorizado” los datos de entrada.

11.1. Resultados del test

Para los experimentos, se ha decidido multiplicar por 4 la fuerza del mando, siguiendo la ecuación 11.

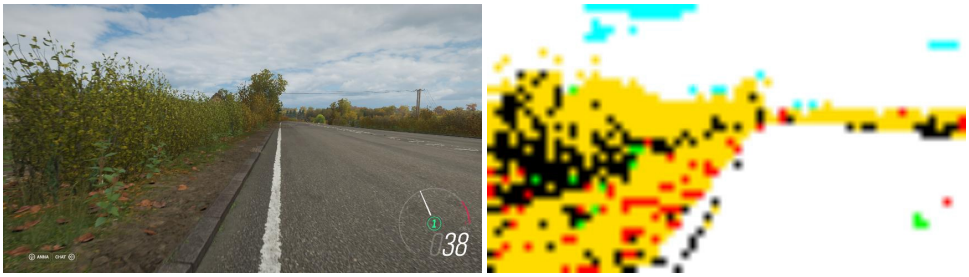
$$volante = ((volante - 0{,}5) \cdot 4) + 0{,}5 \tag{11}$$

Experimento 1

- **500 epochs:** se producen 16 fallos, con una media de 13,78 segundos entre ellos. Pese a este dato, que está en la línea de lo visto hasta el momento, el sistema es capaz de conducir mucho mejor que hasta ahora. Ha sido capaz de girar en todas las curvas, y tan sólo se ha salido cuando ha detectado por un momento alguna que en realidad no exsistía (figura 38). Si bien es cierto que no ha captado la última curva, que es la más cerrada, esto puede deberse a que el hecho de que la calzada se discretizase como negra en lugar de blanca debido a la iluminación (figura 39).

- **5000 epochs:** se han producido 3 fallos, con una media de 82,16 segundos entre ellos. Esto supone una mejora radical respecto a todos los modelos analizados hasta el momento. De hecho, durante los 4 primeros minutos, el sistema ha sido capaz de conducir sin fallo alguno, por el carril izquierdo (cabe recordar que el entorno virtual está basado en Reino Unido) excepto

en una ocasión en la que ha vuelto a él rápidamente. Los tres fallos han sido ocasionados por la curva cerrada a la derecha mencionada anteriormente. Pese a que el modelo la ha detectado, la velocidad ha sido demasiado elevada. Parece, además, que el sistema ha perdido la capacidad de moderar la velocidad, llegando incluso hasta los 70 km/h con facilidad en bastantes ocasiones.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 38: Situación sin curva en la que el modelo ha mostrado intención de girar. Fuente: elaboración propia.

(a) Imagen real. (b) Imagen vista por el sistema.

Figura 39: Situación de curva a la derecha en la que el modelo no ha girado. Fuente: elaboración propia.

En base a los resultados obtenidos hasta el momento, el modelo final escogido será este último: una red neuronal que utiliza el color discretizado en 8 valores distintos, entrenado con 5 minutos y 5000 epochs.

12. Resultados de generalización

A continuación, se comprobará cómo de bien generaliza el modelo en las distintas situaciones descritas en el apartado X. A priori, parece que el hecho de que se haya entrenado con 5 minutos de datos de carretera interurbana podría afectar negativamente al nivel de generalización para vías y situaciones que no se le hayan presentado al sistema durante su entrenamiento.

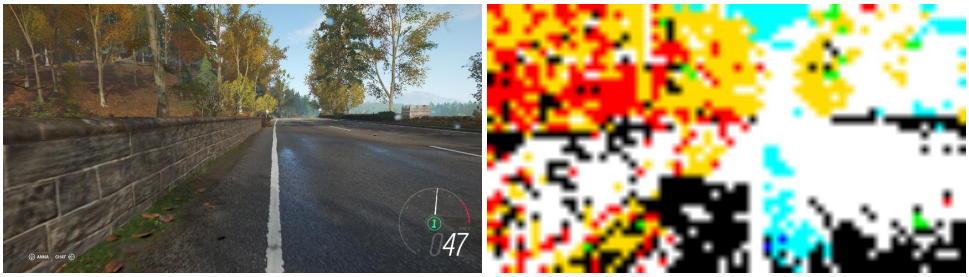
12.1. Conducción por vía interurbana desconocida

Se produce un fallo cada 21,51 segundos de media, con un total de 11 fallos tras los 5 minutos de conducción. El vehículo conduce razonablemente bien, considerando que es una carretera con la que no ha entrenado, pero del mismo tipo que las que aparecían en el conjunto de entrenamiento. La velocidad, por su parte, ha sido muy inestable: ha llegado a ir a 30 km/h en algunos puntos, mientras que en cierto momento ha llegado a acelerar hasta los 90 km/h. Las situaciones en las que el vehículo se ha salido de la calzada ha sido por velocidades excesivas que no le han permitido mantener la trazada, porque el exterior de la carretera también se discretiza como color blanco, o porque el color de la misma no es el esperado.

En la figura 40, se puede observar una situación en la que el vehículo se ha salido de la calzada debido a que el asfalto se ve de color negro producido por una sombra, cuando en el conjunto de entrenamiento la carretera era blanca. En la figura 41 se observa una imagen que es demasiado ruidosa, y donde no se distingue bien la calzada blanca. En la imagen de la figura 42, se observa como el vehículo ha tratado de seguir de frente sin modificar su trayectoria para incorporarse a la vía. Esto se debe, muy probablemente, a la pequeña acera que se ve, que también se discretiza de color blanco, confundiéndose con parte de la carretera.

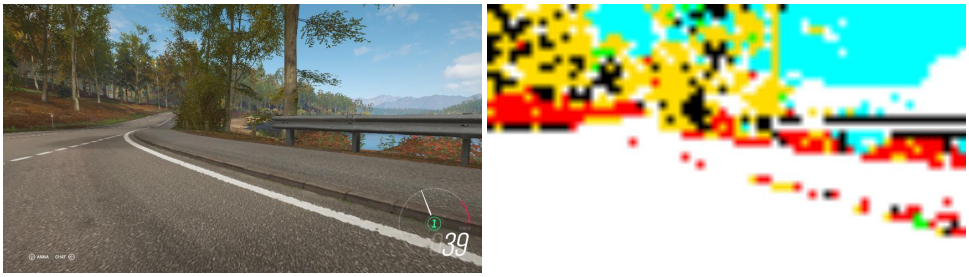
(a) Imagen real. (b) Imagen vista por el sistema.

Figura 40: Situación de calzada con sombra en la que se ha producido un fallo. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 41: Imagen ruidosa que ha confundido al modelo. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 42: Situación de incorporación a otra vía en la que el sistema ha fallado. Fuente: elaboración propia.

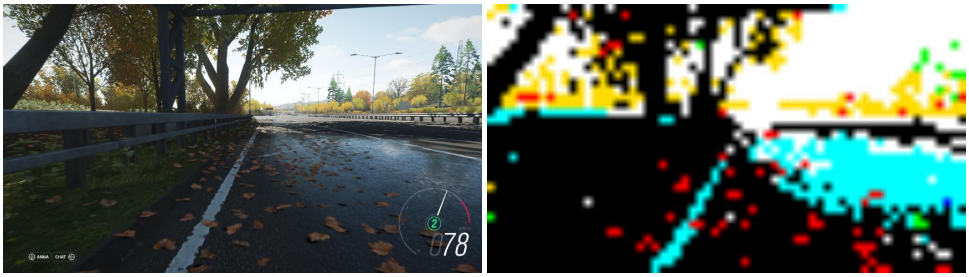
12.2. Conducción por autopista

Se producen un total de 24 fallos, con una media de tiempo entre ellos de 8,70 segundos. El sistema es capaz de discerninr las curvas, y trata de mantenerse lo más a la izquierda posible. Esto es algo positivo, puesto que las normas de circulación obligan por norma general a circular por el carril más exterior. Pese a la multitud de fallos, la mayoría de éstos se han producido por choques contra el guardarraíles, por una cercanía excesiva al borde de la calzada. También se han producido fallos dentro de un túnel, donde la visibilidad queda muy restringida y el modelo tiene problemas. En cuanto a la velocidad, por su parte, ha sido adecuada para la vía, si acaso algo reducida.

En la figura 43 y 44, pueden verse dos ejemplos de los fallos del vehículo al acercarse en demasía al guardarraíles lateral. Esto es porque el vehículo ha aprendido a mantener una cierta distancia con el borde izquierdo en las carreteras convencionales. Dicha distancia resulta insuficiente en autopista y, debido a ello, colisiona con las vallas laterales. En la figura 45, vemos que el sistema no tenía ningún tipo de información para poder circular correctamente por el túnel, lo que ha desembocado en colisiones laterales contra las paredes del túnel en cuatro ocasiones.

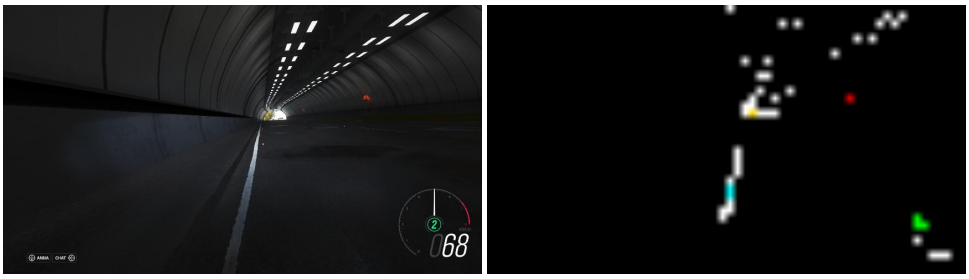
(a) Imagen real. (b) Imagen vista por el sistema.

Figura 43: Situación de guardarraíles próximo que provoca un fallo. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 44: Situación de calzada oscura con guardarraíles próximo que provoca fallo. Fuente: elaboración propia.



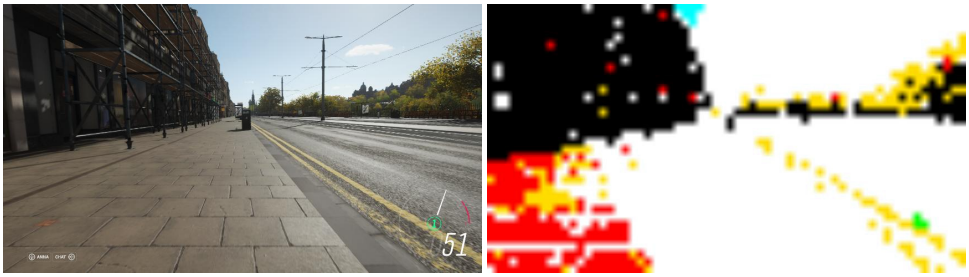
(a) Imagen real. (b) Imagen vista por el sistema.

Figura 45: Tramo de túnel en el que el modeo falla. Fuente: elaboración propia.

12.3. Conducción en vía urbana

Por lo que respecta a la conducción por vías urbanas, durante los 5 minutos de conducción, se han producido 26 fallos, con una media de 6,40 segundos entre ellos. El problema principal radica en la identificación errónea de las aceras amplias como parte de la calzada, lo que lleva al sistema a dirigirse hacia ellas y a circular sobre éstas manteniendo la distancia con los edificios. La velocidad ha sido adecuada en todo momento.

En la figura 46, puede observarse una de las situaciones en las que el vehículo se sube a la acera. Al subirse a ella, el sistema muchas veces no comprendía que debía evitar el edificio y acababa estrellándose contra él. Esto es porque los edificios son normalmente de color negro (figura 47) para el modelo, mientras que en los ejemplos proporcionados sólo ha visto bordes mayoritariamente amarillos o rojos. Además de entrar en conflicto con las aceras, tampoco reconoce correctamente las isletas porque son del mismo color que la calzada (figura 48). En algunas situaciones, hay una línea amarilla en un borde (figura 49) que podría ayudar a un modelo entrenado con imágenes de conducción en poblado a entender que debe circular dejándola a la izquierda, sin invadir la calzada. Sin embargo, puede ser este uno de los motivos que confundía al modelo de 25 minutos, donde la mayor parte del tiempo se circulaba por la parte blanca (para el modelo) de la imagen, y en otras ocasiones se hacía dejando a la izquierda de una línea amarilla. Acabó, aparentemente, por no aprender cómo gestionar ninguna de las dos situaciones.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 46: Situación en la que el modelo se sube a la acera. Fuente: elaboración propia.

(a) Imagen real. (b) Imagen vista por el sistema.

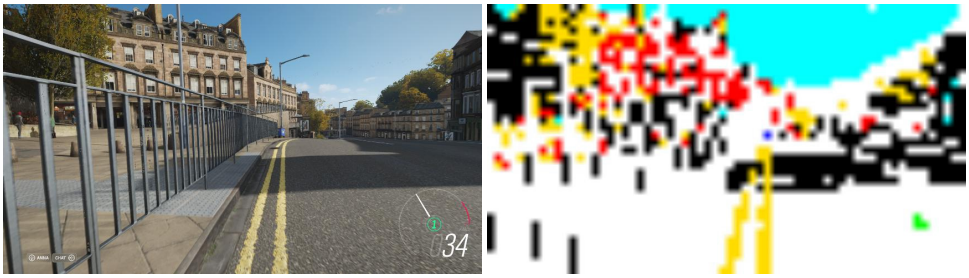
Figura 47: Edificios representados en negro. Fuente: elaboración propia.

12.4. Conducción nocturna

Como se podría esperar a priori, la generalización a la conducción autónoma no funciona en absoluto. Con un MTBF de 4,40 segundos y un total de 37 errores, se trata de la situación en la que peor se comporta el sistema. Lo único que man-

(a) Imagen real. (b) Imagen vista por el sistema.

Figura 48: Situación con isleta que no percibe el sistema. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

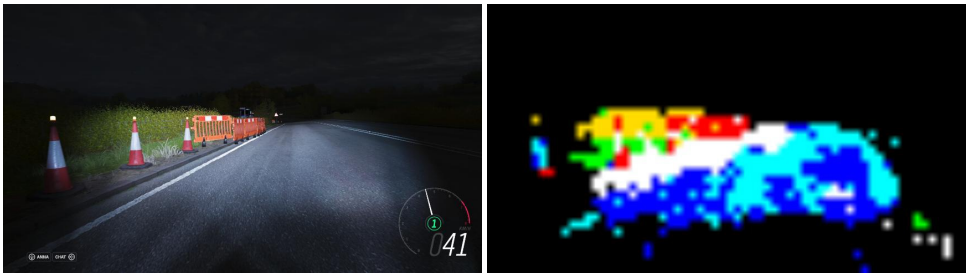
Figura 49: Tramo con línea amarilla a la izquierda. Fuente: elaboración propia.

tiene de forma correcta es la velocidad, que ha sido adecuada en todo momento.

Si se observa la figura 50 o la figura 51, resulta evidente que el modelo no veía nada de lo que éste se esperaba. La calzada blanca no aparece, como tampoco lo hacen de forma clara los bordes amarillos o rojos de los que ha aprendido a separarse correctamente. Así pues, las salidas de la vía son continuas. Además, con esta situación de ”cegera”, el modelo tiene un cierto sesgo a girar hacia la izquierda, pero debido a la opacidad propia de las redes neuronales, es imposible determinar la causa concreta de este comportamiento.

(a) Imagen real. (b) Imagen vista por el sistema.

Figura 50: Calzada azul que confunde al sistema. Fuente: elaboración propia.



(a) Imagen real. (b) Imagen vista por el sistema.

Figura 51: Calzada con varios colores que confunde al sistema. Fuente: elaboración propia.

13. Resultados

Así pues, durante el desarrollo del sistema inteligente, se han creado un total de cinco modelos intermedios, cuyos resultados pueden resumirse de la siguiente forma:

- **Modelo inicial:** una red de dos capas internas con 64 neuronas cada una. El sistema no es capaz de tomar ninguna curva, pese a que a veces muestra la intención de hacerlo. Cuando se le añade una función para multiplicar la fuerza del volante para que *FH4* lo reconozca, parece que existe un sesgo a la derecha, provocando salidas de la calzada continuas del vehículo por ese lado. Además, no es capaz de tomar las curvas a la izquierda. El modelo sí es capaz de mantener una velocidad adecuada, entre los 30 y los 40 km/h.

- **Ampliación de capas y neuronas:** se comprueba si el modelo funciona mejor ampliando el número de capas y el de neuronas en cada una de ellas (cuatro capas internas con 512 neuronas cada una). Los resultados son similares a los del modelo anterior, sin que exista una clara mejora. Así pues, se sigue adelante con la arquitectura simple puesto que no supone diferencia en cuanto a rendimiento pero sí que resulta más sencillo y rápido de entrenar. Parece ser que con los datos proporcionados al modelo, el número de neuronas y de capas no mejora el rendimiento del sistema, probablemente porque éste no tiene información suficiente para imitar la conducción humana.

- **Reducción de ruido:** como uno de los problemas podía ser la variabilidad propia del volante, la variable correspondiente a éste se discretiza a un solo decimal, para ver si esto supone alguna mejora en el rendimiento del sistema. Por desgracia, los resultados no presentan ningún cambio sustancial, por lo que el problema no radica en los constantes cambios en la dirección del volante frutos del control humano.

- **Adición de color:** con un modelo que detecta ocho colores distintos, el sistema ya es capaz de seguir las curvas de forma mínimamente correcta. No

obstante, aún falla demasiado (cada 11 segundos, de media) y parece perder el control de la velocidad, que sube en algunos casos a 70 km/h e impide tomar curvas cerradas.

- **Reducción del conjunto de entrenamiento:** se decide finalmente reducir el tamaño de los datos del conjunto de entrenamiento para comprobar si es capaz de aprender bien a conducir por carreteras convencionales, sin tener en cuenta tantas situaciones distintas ni conducción urbana. Bajo estas condiciones, el modelo aprende extraordinariamente bien, y es capaz de girar en todas las curvas. Además, se mantiene, por lo general, dentro de su carril. Así pues, se escoge este como modelo final.

Tras el desarrollo del sistema inteligente, se comprueba cómo de bien generaliza poniéndolo a prueba en distintas condiciones. Los resultados de estos tests pueden resumirse como sigue:

- **Conducción por vía interurbana desconocida:** el modelo ha aprendido correctamente a conducir por cualquier vía interurbana convencional, tanto con tiempo soleado como con llovizna. Sólo se sale de la calzada en intersecciones en forma de ”T” y cuando el borde de la misma presenta el mismo color que el asfalto.

- **Conducción por autopista:** el vehículo detecta correctamente las curvas, pero por desgracia se aproxima demasiado al borde de la calzada (cosa que ha aprendido y funciona de las vías convencionales), accidentándose contra el quitamiedos. Además, toma muchas veces las salidas de la autopista pues se mantiene pegado lo más a la izquierda posible. El modlo parece, además, conducir más rápido por este tipo de carreteras, y siempre por el carril izquierdo.

- **Conducción por vía urbana:** la conducción en poblado le resulta muy complicada al sistema. Las aceras, que son del mismo color que la calzada, son detectadas como parte de la misma, por lo que el vehículo acaba subiéndose a éstas y accidentándose con el mobiliario urbano. Es probable que con más entrenamiento tanto con datos de conducción urbana como interurbana, el modelo aprendiese a desenvolverse bien en ambas situaciones. También podrían desarrollarse dos redes independientes según el tipo de vía y algún sistema que detectase en qué situación se encuentra el vehículo para activar una u otra. De esa forma, las dos redes independientes no deberían aprender a discernir la conducción en vía interurbana y urbana a la vez.

- **Conducción nocturna:** esta es, sin duda, la situación en la que el sistema funciona peor. Al guiarse por la cámara situada en el frontal del vehículo, cuando anochece no capta nada excepto una pequeña región iluminada por los faros del coche. Sin embargo, los colores cambian respecto a lo aprendido por el modelo cuando son iluminados por las luces del vehículo. Para

solventarlo, o bien se entrena la red con situaciones diurnas y nocturnas al mismo tiempo, o se crean dos redes separadas y se activan en función de la luminosidad. Sin duda, sería interesante ver qué funciona mejor, pero queda fuera del alcance de este proyecto (ver Trabajo futuro).

Se han conseguido cumplir todos los objetivos propuestos en este trabajo (apartado 3.2): se ha desarrollado un sistema inteligente de conducción autónoma capaz de conducir sin salirse de la calzada la mayoría del tiempo, y que por lo general circula por el carril izquierdo. Se han analizado los problemas hallados y las situaciones que provocan un fallo del sistema, y se ha acabado comprobando cómo de bien generaliza el modelo final en distintas situaciones. El sistema es capaz de conducir de forma eficaz en carreteras convencionales interurbanas, pero tiene problemas y requeriría que un humano tomase el control, en ciudad, autopista o situaciones de luminosidad insuficiente.

14. Planificación temporal

En esta sección, se discute la planificación temporal desgrananda en tareas y subtareas, y explicando de qué recursos hace uso cada una de ellas. Además, se habla también de cómo y por qué ha cambiado dicha planificación respecto a la inicialmente planteada (anexo A) en el Curso de Gestión de Proyectos (GEP).

La realización de este proyecto, incluyendo tanto la implementación como la memoria, ha tenido lugar entre el 22 de febrero de 2019 y el 27 de junio de 2019.

14.1. Descripción de las tareas

A contunuación, se describirá cada una de las tareas en las que se ha dividido el presente trabajo:

14.1.1. Adquisición del conocimiento previo

Antes de poder empezar con el diseño de la solución, es necesario adquirir los conocimientos necesarios para poder hacerlo. En esta primera fase del proyecto se incluye la investigación sobre la tecnología a utilizar (aprendizaje por imitación, redes neuronales), la búsqueda de trabajos similares en el campo de la conducción autónoma (tomar consciencia de cuál es el estado del arte) y la selección de los distintos programas y librerías que, a priori, parecen adecuadas para el desarrollo del proyecto.

14.1.2. Toma de contacto y planificación

Por tal de comprobar que las librerías previamente escogidas resultan de utilidad y que todas ellas son compatibles las unas con las otras hay que realizar pequeños tests. En ellos, se ha comprobado que no hay problemas de compatibilidad entre ellas y se ha asegurado de que las librerías escogidas (apartado 4.3) permitirán implementar correctamente el sistema. De igual forma, hay que probar los programas que se vayan a utilizar (apartado 4.3) por tal de verificar que realmente sirven para los propósitos designados. Es fundamental detectar en esta fase cualquier problema de conflictividad entre librerías y programas, puesto que un error no detectado puede significar el retraso de una fase posterior cuando el proyecto ya esté demasiado avanzado como para rectificar. En esta tarea también se incluye la documentación del curso GEP (Curso de gestión de proyectos).

Esta tarea requiere de todas las librerías y los programas mencionados en la sección 4.2, así como el uso del controlador externo (Xbox One Wireless Controller) con el que se emularán los controles del vehículo y mediante los que el experto proporciona los datos.

14.1.3. Reconocimiento de velocidad

FH4 no dispone de ningún tipo de API con la que extraer datos del videojuego. Es por esta razón que la velocidad del coche en cada momento, que junto con la imagen desde el frontal del vehículo representa la entrada de la red neuronal, debe obtenerse mediante visión por computador. El objetivo de esta tarea es implementar un sistema que reconozca los números en pantalla y devuelva la velocidad en cada momento. Esto no es una tarea complicada, pues los números siempre tienen el mismo color (se pueden filtrar del resto de la imagen) y están en la misma posición (se puede averiguar qué número es por simple comparación). Esta actividad se divide en tres subtareas: implementación (donde también se incluye la obtención de los ejemplos de los dígitos con los que comparará el modelo), la comprobación de que el sistema funciona, y la documentación de todo el proceso.

De igual forma que en la tarea anterior, esta también requiere de todas las librerías y programas y el mando de control. De ahora en adelante, todas las tareas harán uso de estos recursos hardware y software excepto que se explicite lo contrario.

14.1.4. Primer prototipo de red

Con el objetivo de acabar de entender el funcionamiento de las distintas librerías listadas en el apartado 4.3, y ver cómo interaccionan y se integran de forma conjunta, primero se desarrollará una red con la misma entrada e igual salida que la que se pretende para el diseño final. La topología de la misma no será un punto clave de este primer prototipo, simplemente se buscará que dada la imagen del frontal y la velocidad, proporcione un estado de la dirección y de los pedales de aceleración y freno, aunque no sea necesariamente correcto y no sea capaz de conducir de forma racional. Esta tarea se divide en tres subtareas: implementación, entrenamiento (recogida de ejemplos del experto) y puesta a prueba del sistema. De ahora en adelante, todas las tareas se dividen de igual forma excepto en las que se especifique lo contrario.

Dado que esta tarea tan sólo existe para habituarse al uso de las distintas librerías y aprender cómo interactúan estas entre ellas, no se documentará en esta memoria. Esto es así porque no forma parte del estudio en sí, si no que corresponde a pequeñas pruebas personales sin interés para este trabajo.

14.1.5. Modelos

Esta tarea comprende todo el desarrollo de los distintos modelos creados hasta dar con el modelo final, estando éste último incluido. En cada uno de las subtareas, se incluyen la implementación, el entrenamiento, el testeo y la documentación asociada.

Las subtareas, y por lo tanto los modelos, son los que siguen:

- **Modelo inicial**
- **Ampliación de capas y neuronas**
- **Reducción de ruido**
- **Adición de color**
- **Reducción de los datos**

14.1.6. Generalización

En esta tarea, se comprobará cómo se desenvuelve el modelo final en las distintas situaciones propuestas en el apartado X. Además, se analizarán las circunstancias en las qué ocurren los fallos y se intentará proporcionar una explicación de por qué ocurre y cómo podría atajarse el problema.

14.1.7. Puesta a punto de la memoria

Se trata de la edición y corrección de esta memoria, así como completar las últimas secciones, de cara a la entrega final.

14.1.8. Reuniones

Por último, también debemos tener en cuenta todas las reuniones que se han llevado a cabo de forma habitual para garantizar el correcto desarrollo del trabajo, así como para resolver los problemas hallados o redefinir el plan de acción ante situaciones inesperadas. Esta tarea no se subdivide en ninguna otra.

Esta última tarea no requiere, en principio, de ningún recurso software o hardware, a menos que la reunión tenga por objeto alguna demostración o implementación conjunta con el director, en cuyo caso serían necesarias las herramientas ya mencionadas.

14.2. Dependencias entre tareas y otras consideraciones

A excepción de las reuniones y el GEP (Curso de Gestión de Proyectos), todas las tareas y subtareas descritas previamente se mencionan en orden cronológico estableciéndose por tanto una dependencia consecutiva entre ellas. Esta jerarquía puede verse de forma clara en la figura 52. La documentación del GEP puede generarse con independencia de cuánto trabajo se ha realizado hasta el momento. Por su parte, las reuniones han tenido lugar en días específicos pero, al no ser relevante esto para este trabajo, se ha decidido omitirlas en el diagrama de Gantt.

14.3. Programación de tareas

En esta sección, se analiza el coste temporal de cada una de las tareas descritas anteriormente, y se presenta un calendario en forma de diagrama de Gantt que ilustra cómo se han distribuido las tareas en el tiempo.

14.3.1. Coste temporal

En la Tabla 1, puede visualizarse el coste temporal, en horas, para cada una de las tareas y sus respectivas subtareas. Se puede observar que, en total, el proyecto ha durado alrededor de las 450 horas. El TFG equivale a 15 créditos ECTS y, según la propia UPC, cada uno de ellos son 30 horas de trabajo (UPC, año desconocido). Si realizamos la operación, tenemos 15 créditos $\cdot \frac{30 \text{ horas}}{1 \text{ crédito}} = 450$ horas, que son exactamente las horas que han sido necesarias para la realización del proyecto.

14.3.2. Diagrama de Gantt

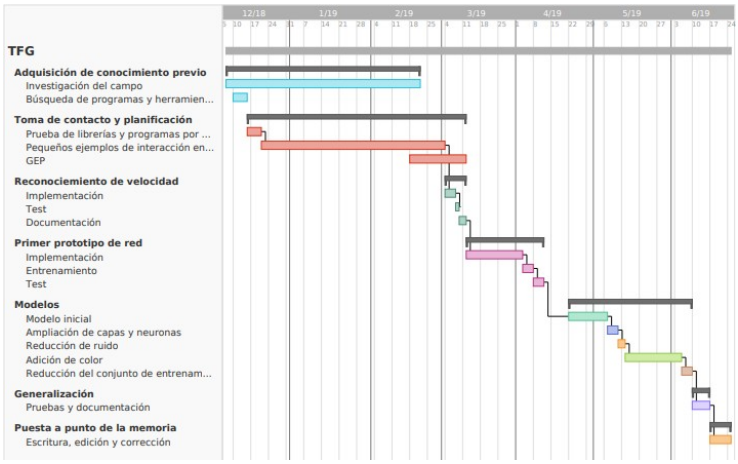


Figura 52: Diagrama de Gantt. Creado mediante *www.teamgantt.com*

Deben hacerse varias consideraciones respecto al diagrama de Gantt mostrado en la figura 52. En primer lugar, se trata de una aproximación del tiempo que se ha tardado en desarrollar cada tarea en función de los días trabajados y las horas trabajadas de media en cada uno de ellos (entre 4 y 5 horas). En segundo lugar, la extensión de cada tarea muestra el inicio y el final de la misma, pero no tiene por qué haberse trabajado todos los días marcados ni el mismo tiempo (por ejemplo, algunos festivos no se han trabajado debido a los desplazamientos vacacionales que estos suelen conllevar). Por último, las reuniones se han dejado fuera en este diagrama puesto que no tendría sentido (ocuparía desde el inicio del proyecto hasta el final del mismo, no aporta información).

Tarea	Tiempo estimado (en horas)
Adquisición de conocimiento previo	
Investigación del campo	30
Búsqueda de programas y herramientas	30
Subtotal	60
Toma de contacto y planificación	
Prueba de librerías y programas por separado	5
Pequeños ejemplos de interacción entre librerías	10
GEP	25
Subtotal	40
Reconociemiento de velocidad	
Implementación	10
Test	5
Documentación	5
Subtotal	20
Primer prototipo de red	
Implementación	30
Entrenamiento	20
Test	10
Subtotal	60
Modelos	
Modelo inicial	60
Ampliación de capas y neuronas	30
Reducción de ruido	20
Adición de color	60
Reducción del conjunto de entrenamiento	20
Subtotal	190
Generalización	
Subtotal	40
Puesta a punto de la memoria	
Subtotal	30
Reuniones	
Subtotal	10
Total	450

Tabla 1: Estimación temporal en horas de las distintas tareas del proyecto.

14.4. Cambios respecto a la planificación inicial

Hay varios aspectos que han sufrido cambios respecto a lo que se proponía en la planificación inicial, disponible en el anexo A. A continuación, se detallara cuáles han sido y se explica brevemente el motivo de los cambios.

14.4.1. Planificación de tareas

En lo que respecta a la división en tareas del proyecto, las tareas ”Red funcional en condiciones favorables”, ”Entrenamiento con distintas condiciones” y ”Finalización” se han sustituido por aquellas tareas que, finalmente, han acabado teniendo lugar. Al principio, el trabajo se planificó teniendo en cuenta sobretodo la parte del desarrollo, y no tanto en el análisis de los errores y el planteamiento de explicaciones y soluciones a los mismos, que es sin duda lo más interesante. Por este motivo, tras las pertinentes reuniones con el director del proyecto, se decidió que se buscaría un sistema funcional de acuerdo con lo dispuesto en el apartado 3.2, analizando los errores de por medio y llegando a la solución óptima a partir de iterar a través de varios modelos. Así pues, en la división por tareas final, se reflejan cuáles han sido las iteraciones que han tenido lugar finalmente.

14.4.2. Tiempo requerido

Pese a que inicialmente se proyectaron unas 500 horas para la realización del trabajo, finalmente han sido sólo 450. No es que el desarrollo del trabajo haya sido más rápido de lo que se pensaba inicialmente, si no que las horas de contingencia que se sumaron por si aparecía algún contratiempo personal o tecnológico no se han acabado usando. Esto significa que el trabajo se ha desarrollado sin problemas dentro del plazo dado. No obstante, sí que se ha terminado unos días después de lo esperado puesto que muchas de las horas disponibles se han ido finalmente a la realización, edición y corrección de la presente memoria los últimos días.

15. Coste del proyecto

15.1. Presupuesto

A continuación, se procede a analizar el coste de este proyecto. Para hacerlo de la forma más precisa posible, en lugar de realizar los cálculos mediante el diagrama de Gantt, que indica qué día empieza y termina cada tarea (apartado 14.3.2), se computarán teniendo en cuenta las horas estimadas para cada una de ellas (tabla 1).

15.1.1. Recursos humanos

Se analizará primero de cuál es el coste derivado de los recursos humanos. Para ello, deben tenerse en cuenta los roles descritos en el apartado 4.1. A continuación, se muestran las siguientes tablas que ayudan al cálculo del coste imputable a los recursos humanos:

Rol	Tareas	Horas por tarea
Cabeza de proyecto	1, 2, 3, 4, 5, 6, 7, 8	60, 40, 20, 60, 190, 40, 30, 10
Experto en Machine Learning	3, 4, 5, 6	20, 60, 190, 40
Programador	1, 2, 3, 4, 5	30, 15, 10, 30, 190
Experto (conductor)	3, 4, 5	10, 20, 190
Tester	3, 4, 5, 6	5, 10, 190, 40

Tabla 2: Tareas en las que participa cada uno de los roles y con cuántas horas en cada una respectivamente.

Rol	Sueldo por hora (en €)	Total de horas	Precio (en €)
Cabeza de proyecto	50	450	22.500
Experto en Machine Learning	70	310	21.700
Programador	30	275	8.250
Experto (conductor)	70	220	15.400
Tester	30	245	7.350
Total			75.200

Tabla 3: Presupuesto destinado a recursos humanos, detallando sueldo y horas trabajadas por cada uno.

Consideraciones:

- El cabeza de proyecto debe estar presente durante todo el proyecto.
- El experto en *Machine Learning* participa de las fases de implementación y entrenamiento (si existe dicha subtarea) de las tareas asignadas.

- Como no hay ”subsubtareas”, no se puede discernir entre las fases de implementación, entrenamiento, test y documentación de los distintos modelos. Por eso, se asume que si un rol participa de alguna forma en la creación de los modelos, lo hace durante toda la subtarea.
- El programador y el tester se asume que son una misma persona, por eso el sueldo idéntico.
- Para estimar el sueldo del cabeza de proyecto y programador (y tester), la cifra se ha obtenido de trabajos previos como sugería la guía del GEP. Para el experto en *Machine Learning*, se ha aproximado la cifra proporcionada por el estudio de *HowMuch* [7].

15.1.2. Recursos software

A continuación, se analizan los costes del software utilizado en este proyecto (los recursos software están explicados en el apartado 4.3 de este documento):

Software	Tareas	Horas por tarea	Total de horas
Python	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
Librerías de Python (1)	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
PyCharm	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
LaTeX (Overleaf)	2, 3, 5, 6, 7	25, 5, 190, 40, 30	290
Forza Horizon 4	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
XOutput2	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
vJoy	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320
Google Chrome	1 (2)	60	60
Windows 10	1, 2, 3, 4, 5, 6, 7	60, 40, 20, 60, 190, 40, 30	440
Photoshop	7	30	30
FastStone Capture	7	30	30
TeamGantt	2, 7	25, 30	55

Tabla 4: Tareas en las que se usa cada software y cuántas horas en cada una.

Software	Precio (en €)	Horas de uso	Amortización (en €) (3)
Python	0	320	0
Librerías de Python (1)	0	320	0
PyCharm	0	320	0
LaTeX (Overleaf)	0	290	0
Forza Horizon 4	53,71	320	94,18 (4)
XOutput2	0	320	0
vJoy	0	320	0
Google Chrome	0	60	0
Windows 10	0	440	0
GIMP	0	30	0
FastStone Capture	0	30	0
TeamGantt	0	55	0
Total			94,18

Tabla 5: Presupuesto destinado a recursos software.

Consideraciones:

- (1) OpenCV, TensorFlow, TFLearn, PIL, xbox360controller, xbox360controllerWrapper, NumPy, librerías estándar de Python.
- (2) Se asume que el navegador Google Chrome sólo se usa durante la investigación. Previsiblemente también se usará para resolver dudas durante el desarrollo o para pequeñas comunicaciones con el director (como correos), pero en total se presumen horas despreciables en cuanto al presupuesto (atendiendo además a que es software libre).
- (3) Para calcular la amortización, se usa la fórmula:

$$\left(\frac{horas\_uso\_proyecto}{horas\_uso\_diario \cdot 365 \cdot años\_vida\_útil}\right) \cdot precio$$

- (4) Forza Horizon 4 es un videojuego, por lo que se le suponen unos 2 años de vida útil y un uso diario medio de 15 minutos (0,25 horas). Este valor por encima del precio significa que se ha usado más de lo normal para un usuario medio, y que se ha amortizado todo su precio.

15.1.3. Recursos hardware

En este apartado se analizan los costes imputables al hardware (la explicación de éstos se encuentra en el apartado 4.2):

Hardware	Tareas	Horas por tarea	Total de horas
Ordenador	1, 2, 3, 4, 5, 6, 7	60, 40, 20, 60, 190, 40, 30	440
Mando Xbox One (inalámbrico)	2, 3, 4, 5, 6	15, 15, 60, 190, 40	320

Tabla 6: Tareas en las que se usa cada componente hardware y cuántas horas en cada una.

Hardware	Precio (en €) (3)	Horas de uso	Amortización (en €)
Ordenador (5)	1100	440	16,58
Mando Xbox One (inalámbrico) (6)	56,99	320	49,96
Total			66,54

Tabla 7: Presupuesto destinado a recursos hardware.

Consideraciones:

- (5) El PC sobre el que se realiza el trabajo está encendido una media de 20 horas cada día del año, y se le supone una vida útil, sin actualizar ningún componente, de 4 años.
- (6) El mando se usa una media de 15 minutos al día (0,25 horas) y tiene una vida útil de 4 años.

15.1.4. Costes indirectos

Por último, se analizarán a continuación los costes indirectos:

- Electricidad: el precio de la electricidad en España ronda los 0,12€/kWh [8], y el ordenador se va a usar 440 horas con una potencia de unos 390W (0,39kW) [9]. Así pues, el coste es de $440 \cdot 0{,}39 \cdot 0{,}12 = 20{,}592$€.
- Internet: el precio medio mensual de una conexión a Internet de hasta 30Mbps (no es necesario más puesto que sólo es usada para buscar información y para comunicaciones con el director) es de 41,3€/mes [10]. Así pues, sabiendo que este proyecto abarca desde febrero hasta junio, abarcará 5 meses: $41{,}3 \cdot 5 = 206{,}5$€.

15.1.5. Presupuesto total

Así pues, el coste total del proyecto es de 75.200 + 0 (FH4 se ha amortizado del todo) + 66,54 + 20,59 + 206,5 = 75.493,63€. Esto es un 24 % más caro de lo que se estimó al principio (anexo B). La razón de esto es la reestructuración de las tareas explicada en el apartado 14, y el hecho de no poder desgranar las ”subsubtareas” de los modelos creados, lo cual permitiría mayor precisión en el cálculo del coste y una clara reducción del mismo.

15.2. Viabilidad

Los trabajos de investigación sobre un campo, no suelen tener por objeto el conseguir un beneficio económico (al menos a corto y medio plazo). Por este mismo motivo, la viabilidad económica de este proyecto no es de especial relevancia. Desde un punto de vista empresarial, las corporaciones realizan este tipo de trabajos de investigación con la esperanza de poder encontrar nuevos procesos o tecnologías que incorporar a nuevos productos, pero muchas veces acaban de forma infructífera y no son rentables. Desde esta óptica, es muy difícil saber si este trabajo hubiese reportado, en una situación real, algún beneficio económico que hiciese viable el proyecto.

16. Sostenibilidad y compromiso social

En esta sección, se analiza la sostenibilidad y el impacto ambiental, social y económico de este proyecto. Para ello, se sigue la matriz de la figura 3 del documento *Mòdul 2.6 - El informe de sostenibilidad.pdf* (Atenea, 2018) proporcionado durante el curso del GEP.

16.1. Impacto ambiental

En primer lugar, se procederá a realizar un análisis en la vertiente ambiental del trabajo, centrado en saber si el proyecto tendrá un impacto positivo o negativo en el entorno y por qué.

En principio, el mayor impacto ambiental producido por el desarrollo de este proyecto, es el gasto energético del equipo informático. Éste tiene un consumo total, al terminar el proyecto, de aproximadamente 171,6 kW (apartado 15.1.4), lo que supone unos 66,1 kg de CO_2 emitidos a la atmósfera. Este es el único impacto producido por este trabajo, puesto que ningún coche está siendo conducido en la realidad ni se está fabricando producto físico alguno cuyos materiales o proceso de fabricación puedan dañar el entorno.

Tampoco existe manera alguna de reducir el impacto ambiental, puesto que la única forma sería hacer que el trabajo durase menos horas y, en consecuencia, se usase menos el ordenador. Por supuesto, esto implicaría que no se habrían trabajado las horas necesarias para hacer de este un trabajo con entidad suficiente como para ser un trabajo final de grado.

En cuanto a escenarios que pudiesen hacer aumentar la huella ecológica del proyecto, sólo existe la posibilidad de que el desarrollo del mismo hubiese tomado más tiempo del pronosticado, aumentándose así el número de horas en las que el ordenador ha estado encendido y, por tanto, consumiendo energía

El hecho de que el sistema inteligente desarrollado y estudiado en este trabajo se entrene y ponga a prueba en un entorno virtual fotorrealista, también ahorra toda la huella ecológica fruto de la conducción de un vehículo real y toda la logística que ello conlleva. Esto es una clara ventaja respecto a proyectos similares que sí dependan de entrenamientos y pruebas en entornos reales.

16.2. Impacto económico

Respecto al impacto económico de este proyecto, se ha visto en el apartado 15, que tiene un coste de 5.493,63 €, que es un 24 % más de lo que inicialmente se estimó (anexo B). Esto se debe, como ya se ha explicado, a que el cálculo realizado no desgrana los costes de la creación de los distintos modelos creados

hasta llegar a la versión final. Al considerar que los roles que participan de alguna forma de estas subtareas lo hacen durante toda su duración, el precio aumenta. Esto se ha hecho así puesto que los distintos roles deben estar presentes en todo momento en calidad de consultores. Esto, que no era así en la estimación inicial, unido al cambio en la planificación de las tareas, añade ese sobre-coste que no está cubierto tampoco por las contingencias que en su momento se añadieron en el GEP.

Los costes de contingencia añadidos al presupuesto inicial (Anexo B), fueron planificados para paliar un alargamiento del proyecto por dificultades técnicas o personales, y no para una reestructuración de las tareas como la que se ha llevado a cabo.

Al igual que en la dimensión económica, el hecho de usar un entorno virtual abarata costes con respecto a un proyecto que entrene y use el sistema inteligente en vehículos reales. Además, tampoco hacen falta permisos especiales (apartado 1.4), por lo que no existen este tipo de gastos adicionales.

16.3. Impacto social

El impacto social de este trabajo es extenso, pues afecta a varios niveles, tanto personalmente como al resto de la sociedad, que puede usar los resultados aquí expuestos para su beneficio de la forma en que se explica a continuación.

De forma personal, este proyecto ha servido para conseguir entender la dificultad de un problema como este. Desarrollar un modelo de conducción autónoma, aunque no sea muy ambicioso, es una tarea complicada. Hay que saber qué datos usar, cómo usarlos y, cuando falla, interpretar lo que está ocurriendo y proponer una solución efectiva al problema. Al realizar un proyecto como este, resulta impactante la cantidad de factores y situaciones que hay que tener en cuenta en la conducción y que muchas veces se pasan por alto.

A nivel de sociedad como tal, este proyecto puede servir a otros investigadores o estudiantes a comprender cuáles son las dificultades a la hora de desarrollar un modelo de inteligencia artificial para la conducción autónoma, y cómo se les puede dar solución. Así pues, colabora en este campo, que acabará aportando a la sociedad una mejora en su seguridad cuando los vehículos autónomos sean una realidad y se elimine el factor humano como causa de los accidentes (Hancock, 2018).

17. Conclusiones

El desarrollo del sistema inteligente para la conducción autónoma diseñado en este trabajo, ha necesitado de cuatro modelos intermedios, incluido el inicial, para conseguir una red capaz de conducir según lo definido en los objetivos (apartado 3.2). Las posibles variantes a estudiar (más capas, más neuronas, más resolución, distinto ratio de aprendizaje...) son tan extensas que resulta imposible explorarlas todas. En este trabajo sólo se han aplicado aquellas que parecían que, a priori, podrían solventar los problemas que se hallaban en cada iteración del modelo.

Los modelos desarrollados, junto con los resultados obtenidos de los mismos, se puede resumir como sigue:

- **Modelo inicial:** se trata de una red con dos capas internas de 64 neuronas cada una. Con esta configuración, y tras un entrenamiento con 25 minutos de datos, 5 de ellos en ciudad y el resto por carreteras convencionales y un fragmento de autopista, el sistema no es capaz de tomar ninguna curva. A priori, podría parecer que es debido a que *FH4* no detecta los leves cambios de dirección que predice el modelo, por lo que se potencian. Sin embargo, esto no solventa nada y, además, pone de manifiesto un problema de sesgo a la derecha, haciendo que el vehículo se salga por dicho lado de la calzada continuamente. La velocidad, sin embargo, si que la mantiene de forma adecuada entre los 30 y los 40 km/h.

- **Ampliación de capas y neuronas:** con una ampliación a cuatro capas, cada una con 512 neuronas, no aporta una mejora en el rendimiento. Por este motivo, finalmente se prosigue con la red más simple.

- **Reducción de ruido:** una de las posibles causas de que el modelo no aprenda correctamente, podía haber sido la gran variabilidad del volante, fruto del impreciso control humano. Sin embargo, tras discretizar el valor de la variable correspondiente al volante, tampoco se aprecia ninguna mejora sustancial.

- **Adición de color:** al añadir el color al modelo, mapeando todos los colores a ocho distintos, el sistema es ya capaz de seguir curvas de forma medianamente correcta. No obstante, aún falla demasiado (cada 11 segundos de media) y pierde el control en curvas cerradas debido a una velocidad excesiva que en algunos casos supera los 70 km/h.

- **Reducción de los datos de entrenamiento:** se prueba finalmente reduciendo el tamaño de los datos de entrenamiento, mostrándole al modelo tan sólo carreteras convencionales. Esto funciona sorprendentemente bien, haciendo que el sistema sea capaz de girar en todas las curvas y mantenerse en el carril izquierdo la mayoría del tiempo. Se escoge este modelo como el definitivo

El hecho de haber entrenado el modelo final sólo con carreteras convencionales debería, en principio, perjudicar en la generalización y extensibildiad del sistema. Tras las pruebas en distintas situaciones, esta hipótesis se vuelve cierta.

- **Conducción por vía interurbana desconocida:** el modelo ha aprendido correctamente a conducir por cualquier vía interurbana convencional, tanto con tiempo soleado como con llovizna. Sólo se sale de la calzada en intersecciones en forma de ”T” y cuando el borde de la misma presenta el mismo color que el asfalto.

- **Conducción por autopista:** el vehículo detecta correctamente las curvas, pero por desgracia se aproxima demasiado al borde de la calzada (cosa que ha aprendido y funciona de las vías convencionales), accidentándose contra el quitamiedos. Además, toma muchas veces las salidas de la autopista pues se mantiene pegado lo más a la izquierda posible. El modlo parece, además, conducir más rápido por este tipo de carreteras, y siempre por el carril izquierdo.

- **Conducción por vía urbana:** la conducción en poblado le resulta muy complicada al sistema. Las aceras, que son del mismo color que la calzada, son detectadas como parte de la misma, por lo que el vehículo acaba subiéndose a éstas y accidentándose con el mobiliario urbano. Es probable que con más entrenamiento tanto con datos de conducción urbana como interurbana, el modelo aprendiese a desenvolverse bien en ambas situaciones. También podrían desarrollarse dos redes independientes según el tipo de vía y algún sistema que detectase en qué situación se encuentra el vehículo para activar una u otra. De esa forma, las dos redes independientes no deberían aprender a discernir la conducción en vía interurbana y urbana a la vez.

- **Conducción nocturna:** esta es, sin duda, la situación en la que el sistema funciona peor. Al guiarse por la cámara situada en el frontal del vehículo, cuando anochece no capta nada excepto una pequeña región iluminada por los faros del coche. Sin embargo, los colores cambian respecto a lo aprendido por el modelo cuando son iluminados por las luces del vehículo. Para solventarlo, o bien se entrena la red con situaciones diurnas y nocturnas al mismo tiempo, o se crean dos redes separadas y se activan en función de la luminosidad. Sin duda, sería interesante ver qué funciona mejor, pero queda fuera del alcance de este proyecto (ver Trabajo futuro).

Así pues, el sistema no es capaz de generalizar más allá de carreteras que del mismo tipo de las que ha visualizado durante el entrenamiento. No obstante, los resultados son prometedores, puesto que si no fuese por el guardarraíles, el modelo habría aprendido también a circular por autopista. Habiendo sido capaz de aprender a manejarse en esta situación, es muy posible que con más entrenamiento del resto de ellas en las que no se desenvuelve correctamente (apartado 18), se consiga

un sistema más robusto a fallos.

Se puede concluir, por tanto, que en un sistema como el aquí desarrollado, se precisaría la asistencia de un humano en la conducción en entornos urbanos, de autopista o de iluminación insuficiente. Se han cumplido todos los objetivos definidos en el apartado 3.2 de forma satisfactoria: se ha construido un sistema inteligente de conducción autónoma mediante aprendizaje por imitación, que es capaz de conducir sin salirse de la vía y que circula la mayor parte del tiempo por el carril izquierdo. Además se han analizado los problemas durante el desarrollo, se han propuesto soluciones a los mismos, y se ha comprobado finalmente cómo de bien generaliza el modelo final.

18. Trabajo futuro

En este trabajo se han explorado varias posibilidades durante la creación del modelo, pero otras muchas han quedado fuera por falta de tiempo y porque exceden el ámbito del proyecto. A continuación, se presentan las principales líneas de investigación desde las que podría continuar el desarrollo del trabajo.

- **Comprobación de hipótesis:** al final, ha sido una reducción del tamaño de los datos de entrenamiento lo que ha hecho que el modelo conduzca de forma correcta por carreteras convencionales. Con más datos, quizá veía tantas situaciones distintas en tan poco tiempo que era incapaz de aprender a conducir bien en ninguna de ellas. Por ello, sería interesante comprobar si esto es así, mediante el entrenamiento de la red con muchos más datos de cada tipo de conducción (interurbana, urbana, por autopista, nocturna, etc).

- **Modificaciones:** se han realizado muchas modificaciones al modelo inicial propuesto, pero el número de posibilidades explota tan pronto que es imposible abarcar todas ellas. Por ejemplo, habría que probar si un aumento en la resolución lleva asociada una mejoría en el rendimiento del sistema, o si el cambio de hiper-parámetros como el ratio de aprendizaje supone un entrenamiento que lleve a una mejor conducción. Sería también interesante saber si una única red entrenada con datos diurnos y nocturnos a la vez, aprendería a conducir bien en ambos momentos del día.

- **Generalización:** además de las pruebas de generalización realizadas, sería muy interesante comprobar, por ejemplo, cómo de bien se desenvuelve el sistema en las distintas estaciones meteorológicas. El modelo final usa el color para orientarse, y la mayoría de las veces la calzada es blanca. Así pues, todo apunta a que en invierno, por ejemplo, el sistema tendría problemas. Así mismo, habría que ver cómo de bien funciona el modelo en condiciones meteorológicas adversas como niebla, lluvia o nevada intensa.También sería de gran interés ver si se adapta a otros modelos de coche, o incluso si puede usarse el mismo sistema para otra clase de vehículos como camiones o motocicletas.

19. Justificación de competencias

A continuación, se listan las competencias técnicas trabajadas en este proyecto, junto con una breve descripción de cómo se han desarrollado dentro del mismo.

- **CCO 1.1. - Evaluar la complejidad computacional de un problema, conocer las estrategias algorítmicas que puedan conducir a su resolución, y recomendar, desarrollar e implementar la que garantice el mejor rendimiento de acuerdo con los requisitos establecidos:** en este trabajo, se desarrolla un sistema inteligente de conducción automática a través de redes neuronales, y se implementan varios modelos hasta llegar al que cumple con los requisitos u objetivos propuestos (apartado 3.2).

- **CCO 2.1. - Demostrar conocimiento de los fundamentos, de los paradigmas y de las técnicas propias de los sistemas inteligentes, y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen estas técnicas en cualquier ámbito de aplicación:** precisamente la tarea de este proyecto es diseñar un sistema inteligente y analizarlo durante y tras su desarrollo.

- **CCO 2.2. - Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano de una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente en los que están relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes:** en el trabajo se usan redes neuronales, que es una forma de adquirir conocimiento que trata de emular el cerebro humano. Además, su uso es para una tarea de percepción, la de la cámara situada en el frontal del vehículo.

- **CCO 2.3. - Desarrollar y evaluar sistemas interactivos y de presentación de la información compleja, y su aplicación en la resolución de problemas de diseño de interacción persona computador:** en el proyecto, se desarrolla un sistema interactivo mediante el cual el experto conductor genera los datos de ejemplo mediante el uso de un mando para emular un volante de coche.

- **CCO 2.4. - Demostrar conocimiento y desarrollar técnicas de aprendizaje computacional; diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las que se dedican a la extracción automática de información y conocimiento a partir de grandes volúmenes de datos:** se trata de otro de los objetivos del trabajo, que es el de desarrollar un sistema que use aprendizaje mediante imitación para la conducción autónoma de vehículos.

Referencias

[1] Atenea (2018). *El infome de sostenibilidad del TFG* [En línea]. Disponible en: https://atenea.upc.edu/pluginfile.php/2669389/mod_folder/content/0/M%C3%B2dul%202.6%20-%20El%20informe%20de%20sostenibilidad%202018.pdf?forcedownload=1 (Accedido: 1 de marzo de 2019).

[2] Avinash Sharame, V. (2017). *Understanding Activation Functions in Neural Networks* [En línea]. Disponible en: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0 (Accedido: 2 de mayo de 2019).

[3] Baomar, J. Bentley. (2018). *Autonomous Navigation and Landing of Airliners Using Artificial Neural Networks and Learning by Imitation* [En línea]. Disponible en: http://www0.cs.ucl.ac.uk/staff/H.Baomar/files/RP3.pdf (Accedido: 20 de noviembre de 2018).

[4] Cámara de Zaragoza (2012). *Cálculo automático de emisiones totales en relación a los consumos energéticos de sus instalaciones* [En línea]. Disponible en: https://www.camarazaragoza.com/wp-content/uploads/2012/10/calculoemisiones.xls (Accedido: 11 de marzo de 2019).

[5] CBInsights (2018). *46 Corporations Working On Autonomous Vehicles* [En línea]. Disponible en: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/ (Accedido: 22 de febrero de 2019).

[6] Codevilla et al (2018). *End-to-end Driving via Conditional Imitation Learning* [En línea]. Disponible en: http://vladlen.info/papers/conditional-imitation.pdf (Accedido: 20 de febrero de 2019).

[7] DGT (2004). *Real Decreto Legislativo 8/2004, de 29 de octubre, por el que se aprueba el texto refundido de la Ley sobre responsabilidad civil y seguro en la circulación de vehículos a motor* [En línea]. Disponible en: https://www.boe.es/buscar/pdf/2004/BOE-A-2004-18911-consolidado.pdf (Accedido: 22 de junio de 2019).

[8] DGT (2015) [A]. *Real Decreto Legislativo 6/2015, de 30 de octubre, por el que se aprueba el texto refundido de la Ley sobre Tráfico, Circulación de Vehículos a Motor y Seguridad Vial* [En línea]. Disponible en: https://www.boe.es/buscar/pdf/2015/BOE-A-2015-11722-consolidado.pdf (Accedido: 22 de junio de 2019).

[9] DGT (2015) [B]. *Tráfico establece el marco para la realización de pruebas con vehículos de conducción automatizada en vías abiertas a la*

circulación [En línea]. Disponible en: http://www.dgt.es/es/prensa/notas-de-prensa/2015/20151116-trafico-establece-marco-realizacion-pruebas-vehiculos-conduccion-automatizada-vias-abiertas-circulacion.shtml (Accedido: 22 de junio de 2019).

[10] DGT, a través de Studylib (2015). *Autorización de pruebas o ensayos de investigación realizados con vehículos de conducción automatizada en vías abiertas al tráfico en general* [En línea]. Disponible en: https://studylib.es/doc/5194724/autorizaci%C3%B3n-de-pruebas-o-ensayos-de-investigaci%C3%B3n-realiz... (Accedido: 22 de junio de 2019).

[11] García Álvarez, R. M. (Año desconocido). *Los niveles de la conducción autónoma* [En línea]. Disponible en: https://www.cea-online.es/blog/213-los-niveles-de-la-conduccion-autonoma (Accedido: 22 de febrero de 2019).

[12] Hancock, P. (2018). *¿Son realmente más seguros los coches autónomos que las personas al volante?* [En línea]. Disponible en: https://www.xataka.com/vehiculos/son-realmente-mas-seguros-los-coches-autonomos-que-las-personas-al-volante (Accedido: 11 de marzo de 2019).

[13] HowMuch (2018). *These are the Hottest Jobs in U.S. in 2018: Lots of Openings and Dream Salaries* [En línea]. Disponible en: https://howmuch.net/articles/there-are-the-hottest-jobs-in-us-2018-lots-of-opening-and-dream-salaries (Accedido: 9 de marzo de 2019).

[14] Matplotlib (2017). *The Matplotlib API* [En línea]. Disponible en: https://matplotlib.org/2.1.1/api/ (Accedido: 11 de junio de 2019).

[15] OuterVision (Año desconocido). *OuterVision Power Supply Calculator* [En línea]. Disponible en: https://outervision.com/power-supply-calculator (Accedido: 10 de marzo de 2019).

[16] Prego, C. (2019). *Qué dice la legislación española sobre los coches autónomos: una instrucción y muchas incógnitas* [En línea]. Disponible en: https://www.xataka.com/vehiculos/que-dice-legislacion-espanola-coches-autonomos-instruccion-muchas-incognitas (Accedido: 22 de junio de 2019).

[17] Pypi (año desconocido). *xbox360controller* [En línea]. Disponible en: https://pypi.org/project/xbox360controller/ (Accedido el: 11 de febrero de 2019).

[18] rFpro (2018). *World first platform enables the test and training of autonomous vehicles in simulation* [En línea]. Disponible en: https://www.adslzone.net/2016/04/29/precio-internet-espana-los-10-paises-mas-caros-europa/ (Accedido: 11 de marzo de 2019).

[19] Tarifaluzhora (Año desconocido). *Precio de la luz por horas* [En línea]. Disponible en: https://tarifaluzhora.es/ (Accedido: 10 de marzo de 2019).

[20] TechRepublic (2016). *Autonomous driving levels 0 to 5: Understanding the differences* [En línea]. Disponible en: https://www.techrepublic.com/article/autonomous-driving-levels-0-to-5-understanding-the-differences/ (Accedido: 22 de febrero de 2019).

[21] TFLearn (Año desconocido). *TFLearn: Deep learning library featuring a higher-level API for TensorFlow* [En línea]. Disponible en: http://tflearn.org/ (Accedido: 2 de mayo de 2019).

[22] UPC (Año desconocido). *UPC: Preguntas más frecuentes sobre los estudios de grado* [En línea]. Disponible en: https://www.upc.edu/es/grados/acceso-y-admision/faqs-estudios-grado (Accedido: 3 de marzo de 2019).

[23] Valero, C. (2016). *El precio de Internet en España entre los 10 más caros de Europa por países* [En línea]. Disponible en: https://www.adslzone.net/2016/04/29/precio-internet-espana-los-10-paises-mas-caros-europa/ (Accedido: 10 de marzo de 2019).

Anexos

Anexo A - Planificación inicial

En este apartado, se analizarán cada una de las tareas que componen el proyecto, dándose una estimación del coste temporal de las mismas, proporcionando también un calendario de ejecución. Hay que tener en cuenta que, al optar por una metodología Agile (en concreto XP), es muy probable que, en caso de encontrar obstáculos, el calendario y la asignación temporal aquí dispuesta cambien sustancialmente para adaptarse a ellos.

La realización de este proyecto (memoria e implementación) comenzó el día 22 de febrero de 2019, y tiene previsto finalizar el día 21 de junio de 2019.

Descripción de las tareas

A continuación veremos las principales tareas en las que se divide el proyecto, describiendo brevemente cada una de ellas.

Adquisición de conocimiento previo

Antes de poder empezar con el diseño de la solución, es necesario adquirir los conocimientos necesarios para poder hacerlo. En esta primera fase del proyecto se incluye la investigación sobre la tecnología a utilizar (aprendizaje por imitación, redes neuronales), la búsqueda de trabajos similares en el campo de la conducción autónoma (tomar consciencia de cuál es el estado del arte) y la selección de los distintos programas y librerías que, a priori, parecen adecuadas para el desarrollo del proyecto.

Toma de contacto y planificación

Para cerciorarnos de que las librerías previamente escogidas nos resultan de utilidad, y que todo funciona entre sí, hay que realizar pequeños tests. En ellos, comprobaremos que no hay problemas de compatibilidad y nos aseguraremos de que tienen las características que son necesarias para poder implementar correctamente el sistema. De igual forma, hay que probar los programas a utilizar y, de nuevo, asegurarnos de que realmente sirven para nuestros propósitos. Es fundamental detectar en esta fase cualquier problema de conflictividad entre librerías y programas, puesto que un error no detectado puede significar el retraso de una fase posterior cuando el proyecto ya esté demasiado avanzado como para rectificar. En esta tarea también se incluye la documentación del curso GEP (Curso de gestión de proyectos).

Esta tarea requiere de todas las librerías y los programas mencionados en el apartado 4.3, así como el uso del controlador externo (Xbox One Wireless Controller)

con el que se emularán los controles del vehículo y mediante los que el experto proporciona los datos.

Reconocimiento de velocidad

FH no dispone de ningún tipo de API con la que extraer datos del videojuego. Es por esta razón que la velocidad del coche en cada momento, que junto con la imagen desde el frontal del vehículo representa la entrada de la red neuronal, debe obtenerse mediante visión por computador. El objetivo de esta tarea es implementar un sistema que reconozca los números en pantalla y devuelva la velocidad en cada momento. Ésto no debería ser una tarea complicada, pues los números siempre tienen el mismo color (se pueden filtrar del resto de la imagen) y están en la misma posición (se puede averiguar qué número es por simple comparación). Esta actividad se divide en tres subtareas: implementación (donde también se incluye la obtención de los ejemplos de los dígitos con los que comparará el modelo), la comprobación de que el sistema funciona, y la documentación de todo el proceso.

De igual forma que en la tarea anterior, esta también requiere de todas las librerías y programas y el mando de control. De ahora en adelante, todas las tareas harán uso de estos recursos hardware y software excepto que se explicite lo contrario.

Primer prototipo de red

Con el objetivo de acabar de entender el funcionamiento de las distintas librerías explicadas con anterioridad, y ver cómo interaccionan juntas, primero se desarrollará una red con la misma entrada e igual salida que la que se pretende para el diseño final. La topología de la misma no será un punto clave de este primer prototipo, simplemente se buscará que dada la imagen del frontal y la velocidad, proporcione un estado de la dirección y de los pedales de aceleración y freno, aunque no sea necesariamente correcto y no sea capaz de conducir de forma racional. Esta tarea se divide en cuatro subtareas: implementación, entrenamiento (recogida de ejemplos del experto), puesta a prueba del sistema y documentación. De ahora en adelante, todas las tareas se dividen de igual forma excepto en las que se especifique lo contrario.

Red funcional en condiciones favorables

Tras el desarrollo del primer prototipo, éste se entrenará con imágenes diurnas en un entorno delimitado y con condiciones de vía similares. El objetivo de esta tarea es investigar qué topología (número de capas de la red, número de neuronas por capa, conexión entre las mismas) es más efectiva. No se conoce mejor forma de determinar estos parámetros que a prueba y error, basándose únicamente en experiencias de investigaciones previas. Aquí será crucial la experiencia del director en este campo, por lo que se pronostican abundantes reuniones para ajustar los

parámetros mencionados.

Al acabar esta etapa, se espera tener un sistema que conduzca de forma autónoma sin salirse de la vía (o haciéndolo en las mínimas ocasiones posibles) en condiciones de poco tráfico y buena iluminación.

Entrenamiento con distintas condiciones

Una vez acabada la etapa previa, el siguiente objetivo es que el sistema sea capaz de manejar el vehículo en condiciones algo más desfavorables: por la noche, con lluvia, con nieve, o con reflejos en la carretera (pavimento húmedo), entre otros. La forma de conseguir esto es, en principio, entrenando el modelo con más ejemplos del experto desenvolviéndose en las situaciones que queremos que estén contempladas por el sistema. Adicionalmente, quizá se deban aplicar algunas técnicas de visión por computador aún por determinar, o modificar la topología de la red.

Finalización

Antes de dar por acabado el proyecto, sería deseable tener algún tipo de interfaz para poder realizar las demostraciones de forma sencilla e intuitiva, sin requerir ejecuciones desde terminal y el cambio de parámetros desde la misma. Para el día de la defensa del trabajo, también sería conveniente tener grabaciones de muestra con el comportamiento del sistema, en caso de no contar con el equipo necesario para ejecutarlo durante la presentación. Esta última fase, tiene por objeto el desarrollo de dicha interfaz, las grabaciones, y cualquier aspecto que deba acabar de pulirse de cara a la entrega final. Esta tarea no se subdivide como las últimas tres, sino que lo hace en tres diferentes: una primera fase de pulido (acabar de ajustar el sistema), la creación de la pequeña interfaz y las pruebas finales (asegurarnos de que todo funciona como se espera).

Reuniones

Por último, también debemos tener en cuenta todas las reuniones que se harán de forma habitual para garantizar el correcto desarrollo del trabajo, así como para resolver los problemas hallados o redefinir el plan de acción ante situaciones inesperadas. Esta tarea no se subdivide en ninguna otra.

Esta última tarea no requiere, en principio, de ningún recurso software o hardware, a menos que la reunión tenga por objeto alguna demostración o implementación conjunta con el director, en cuyo caso serían necesarias las herramientas ya mencionadas.

Dependencias entre tareas y otras consideraciones

A excepción de las reuniones, todas las tareas anteriormente descritas aparecen en orden cronológico, por lo que se establece una dependencia consecutiva entre ellas. Las reuniones con el director son una tarea independiente del resto, que se ejecutará cuando sea necesario o así se acuerde con el profesor. Por su parte, la documentación, parte fundamental del desarrollo del proyecto, no se ha considerado como una tarea en sí y, en cambio, se ha establecido como una subtarea de aquellas que lo requieren (ver Tabla 8).

Programación de tareas

En esta sección, presentaremos en primer lugar una estimación de los costes temporales de cada una de las tareas presentadas anteriormente, así como un calendario en forma de diagrama de Gantt.

Estimación temporal

En la Tabla 8, encontramos una estimación temporal, en horas, para cada una de las tareas y sus respectivas subtareas. Se puede observar que, en total, la duración estimada del proyecto es de 500 horas. El TFG equivale a 15 créditos ECTS y, según la propia UPC, cada uno de ellos son 30 horas de trabajo [6]. Si realizamos la operación, tenemos 15 créditos $\cdot \frac{30 \text{ horas}}{1 \text{ crédito}} = 450$ horas, que son 50 horas menos de lo estimado. Esta diferencia, como se detallará en el apartado 5.4, responde a tiempos de contingencia ante posibles percances.

Diagrama de Gantt

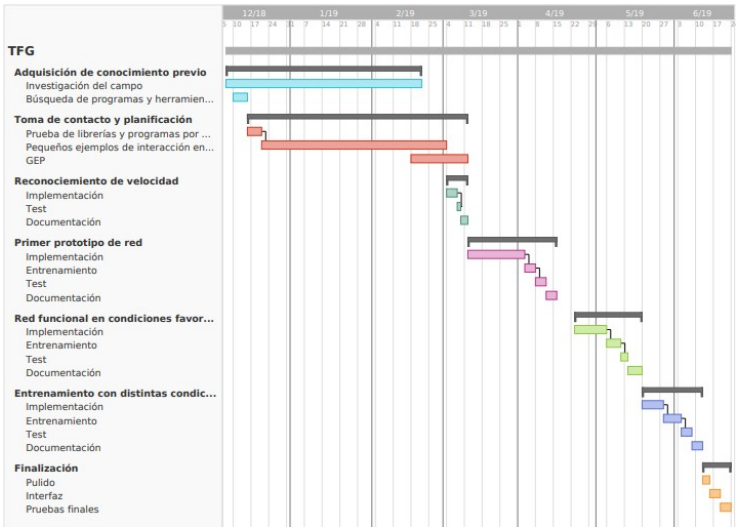


Figura 53: Diagrama de Gantt. Creado mediante *www.teamgantt.com*

Deben hacerse varias consideraciones respecto al diagrama de Gantt mostrado en la Figura 53. En primer lugar, se trata de una aproximación y podría cambiar de forma sustancial durante el transcurso del proyecto, pese a que la intención es que no lo haga. En segundo lugar, la extensión de cada tarea muestra el inicio y el final de la misma, pero no tiene por qué trabajarse todos los días marcados ni el mismo tiempo (por ejemplo, los festivos es probable que no se trabaje debido a periodos vacacionales y los desplazamientos que estos suelen conllevar). Debe contarse una media de 4 a 5 horas por día trabajado. En tercer lugar, es común en el ámbito del *Machine Learning* que las subtareas de implementación, entrenamiento y comprobación sean un proceso cíclico y no secuencial como se muestra en el diagrama. En cuarto lugar, puede observarse que la documentación no tiene dependencias entre subtareas, y esto es porque ésta puede realizarse en paralelo con el resto o bien, como se ha optado en el diagrama, al final de cada tarea. No obstante, de ser necesario para agilizar tiempos, se puede documentar mientras se realiza alguna de las otras subtareas. Por último, las reuniones se han dejado fuera en este diagrama puesto que no tendría sentido (ocuparía desde el inicio del proyecto hasta el final del mismo, no aporta información).

Alternativas y plan de acción

El objetivo principal es seguir el calendario proporcionado por el diagrama de Gantt en la medida de lo posible, realizando las modificaciones que sean necesarias para conseguir superar los posibles problemas con los que nos topemos a lo largo del desarrollo del proyecto. No obstante, todas las estimaciones temporales de la Tabla 8 y, por consiguiente, del diagrama de Gantt de la Figura 53, están realizadas teniendo en cuenta el caso peor. Esto significa que, para cada duración de cada tarea y subtarea, se ha añadido un plus de horas de contingencia para intentar paliar los efectos de un eventual problema durante su realización. Por tanto, el proyecto debería estar acabado antes de la fecha señalada si no aparece ningún contratiempo.

Como se apuntaba en apartados anteriores, existen dos principales obstáculos que podrían poner en peligro la finalización a tiempo del proyecto. Para ambos se ha intentado ofrecer una solución a priori:

- **Desconocimiento tecnológico:** la implementación de los distintos sistemas podría verse seriamente afectada si se desconoce el funcionamiento de las diferentes librerías y programas a utilizar. Peor incluso, podríamos descubrir demasiado tarde que cierto software no permite realizar la tarea para la que estaba designado. Con el fin de reduir este riesgo todo lo posible, y como bien se aprecia en la estimación de tiempos (Tabla 8) y en el diagrama de Gantt (Figura 53), se ha destinado el tiempo previo al comienzo del proyecto como tal a investigar la tecnología disponible, a buscar las librerías y programas adecuados para nuestras necesidades, y a probar que todas funcionan correctamente y son de utilidad para este trabajo.

- **Falta de tiempo:** un fallo tecnológico, una técnica que no funciona cómo se esperaba o una enfermedad, entre otros, podrían poner en serio riesgo la ejecución del proyecto. Por esta razón, y como bien se ha detallado, se han asignado unas horas de contingencia a la planificación temporal. De esta forma, estas horas extra pueden usarse en la resolución de los distintos problemas que puedan ir apareciendo.

Tarea	Tiempo estimado (en horas)	
Adquisición de conocimiento previo		
Investigación del campo		30
Búsqueda de programas y herramientas		30
Subtotal	60	
Toma de contacto y planificación		
Prueba de librerías y programas por separado		5
Pequeños ejemplos de interacción entre librerías		10
GEP		25
Subtotal	40	
Reconociemiento de velocidad		
Implementación		10
Test		5
Documentación		5
Subtotal	20	
Primer prototipo de red		
Implementación		30
Entrenamiento		10
Test		10
Documentación		10
Subtotal	60	
Red funcional en condiciones favorables		
Implementación		30
Entrenamiento		30
Test		30
Documentación		10
Subtotal	100	
Entrenamiento con distintas condiciones		
Implementación		30
Entrenamiento		50
Test		50
Documentación		20
Subtotal	150	
Finalización		
Pulido		20
Interfaz		10
Pruebas finales		20
Subtotal	50	
Reuniones		
Subtotal	20	
Total	500	

Tabla 8: Estimación temporal en horas de las distintas tareas del proyecto.

Anexo B - Presupuesto inicial

A continuación, procederemos a analizar el presupuesto para este proyecto. Para realizarlo de la forma más precisa posible, en lugar de hacer los cálculos mediante el diagrama de Gantt, que indica qué día empieza y termina cada tarea, se computarán teniendo en cuenta las horas estimadas para cada una de ellas.

Recursos humanos

Empezaremos hablando primero de cuál es el coste derivado de los recursos humanos. Para ello, debemos saber cuáles son los roles que participan en el proyecto y de qué tareas se encargan. A continuación, describiremos brevemente cada uno de los roles que participarán en el trabajo:

- **Cabeza de proyecto:** es el líder, está presente en todo momento.
- **Experto en Machine Learning:** participa de todas las fases de implementación y entrenamiento.
- **Programador:** se encarga de escribir el código, aconsejado por el experto en Machine Learning.
- **Experto (conductor):** la persona encargada de crear los ejemplos a partir de los cuales aprenderá el sistema.
- **Tester:** realiza las pruebas necesarias para comprobar la precisión del sistema.

A continuación, tenemos las siguientes tablas para ayudar al cálculo presupuestario imputable a los recursos humanos:

Rol	Tareas	Horas por tarea
Cabeza de proyecto	1, 2, 3, 4, 5, 6, 7, 8	60, 40, 20, 60, 100, 150, 50, 20
Experto en Machine Learning	3, 4, 5, 6	10, 40, 60, 80
Programador	1, 2, 3, 4, 5, 6, 7	60, 40, 15, 50, 70, 100, 50
Experto (conductor)	3, 4, 5, 6	10, 10, 30, 50
Tester	3, 4, 5, 6	5, 10, 30, 50

Tabla 9: Tareas en las que participa cada uno de los roles y con cuántas horas en cada una respectivamente.

Rol	Sueldo por hora (en)	Total de horas	Precio (en)
Cabeza de proyecto	50	500	25.000
Experto en Machine Learning	70	190	13.300
Programador	30	415	12.450
Experto (conductor)	70	100	70.00
Tester	30	95	2.820
Total			60.570

Tabla 10: Presupuesto destinado a recursos humanos, detallando sueldo y horas trabajadas por cada uno.

Consideraciones:

- El cabeza de proyecto debe estar presente durante todo el proyecto.
- El experto en *Machine Learning* participa de las fases de implementación y entrenamiento (si existe dicha subtarea) de las tareas asignadas.
- El programador y el tester se asume que serán una misma persona, por eso el sueldo idéntico.
- Para estimar el sueldo del cabeza de proyecto y programador (y tester), la cifra se ha obtenido de trabajos previos como sugería la guía del GEP. Para el experto en *Machine Learning*, se ha aproximado la cifra proporcionada por el estudio de *HowMuch* [7].

Recursos software

A continuación, analizaremos los costes del software utilizado en este proyecto (los recursos software están explicados en el apartado 4.3 de este documento):

Software	Tareas	Horas por tarea	Total de horas
Python	2, 3, 4, 5, 6, 7	15, 10, 40, 60, 80, 30	235
Librerías de Python (1)	2, 3, 4, 5, 6, 7	15, 10, 40, 60, 80, 30	235
PyCharm	2, 3, 4, 5, 6, 7	15, 10, 40, 60, 80, 30	235
LaTeX (Overleaf)	2, 3, 4, 5, 6	25, 5, 10, 10, 20	70
Forza Horizon 4	3, 4, 5, 6, 7	5, 10, 30, 50, 20	115
XOutput2	2, 3, 4, 5, 6, 7	15, 10, 40, 60, 80, 30	235
vJoy	2, 3, 4, 5, 6, 7	15, 10, 40, 60, 80, 30	235
Google Chrome	1 (2)	60	60
Windows 10	1, 2, 3, 4, 5, 6, 7	60, 40, 20, 60, 100, 150, 50	480

Tabla 11: Tareas en las que se usa cada software y cuántas horas en cada una.

Software	Precio (en €)	Horas de uso	Amortización (en €) (3)
Python	0	235	0
Librerías de Python (1)	0	235	0
PyCharm	0	235	0
LaTeX (Overleaf)	0	70	0
Forza Horizon 4	53,71	115	33,85 (4)
XOutput2	0	235	0
vJoy	0	235	0
Google Chrome	0	60	0
Windows 10	0	480	0
Total			33,85

Tabla 12: Presupuesto destinado a recursos software.

Consideraciones:

- (1) OpenCV, TensorFlow, TFLearn, PIL, xbox360controller, xbox360controllerWrapper, NumPy, librerías estándar de Python.
- (2) Se asume que el navegador Google Chrome sólo se usa durante la investigación. Previsiblemente también se usará para resolver dudas durante el desarrollo o para pequeñas comunicaciones con el director (como correos), pero en total se presumen horas despreciables en cuanto al presupuesto (atendiendo además a que es software libre).
- (3) Para calcular la amortización, se usa la fórmula:

$$\left(\frac{horas\_uso\_proyecto}{horas\_uso\_diario \cdot 365 \cdot años\_vida\_útil}\right) \cdot precio$$

- (4) Forza Horizon 4 es un videojuego, por lo que se le suponen unos 2 años de vida útil y un uso diario medio de 15 minutos (0,25 horas).

Recursos hardware

Continuaremos ahora analizando los costes imputables al hardware (la explicación de éstos se encuentra en el apartado 4.2 de este documento):

Hardware	Tareas	Horas por tarea	Total de horas
Ordenador	1, 2, 3, 4, 5, 6, 7	60, 40, 20, 60, 100, 150, 50	480
Mando Xbox One (inalámbrico)	2, 3, 4, 5, 6, 7	15, 15, 20, 60, 100, 20	230

Tabla 13: Tareas en las que se usa cada componente hardware y cuántas horas en cada una.

Software	Precio (en €) (3)	Horas de uso	Amortización (en €)
Ordenador (5)	1100	480	18,82
Mando Xbox One (inalámbrico) (6)	56,99	230	35,91
Total			54,73

Tabla 14: Presupuesto destinado a recursos hardware.

Consideraciones:

- (5) El ordenador está compuesto por las sigueintes piezas principales: placa base MSI B350 Tomahawk, procesador AMD R7 1700, memoria RAM Corsair DDR4 16GB y gráfica Asus GTX 1060 6GB GDDR5. El PC sobre el que se realiza el trabajo está encendido una media de 20 horas cada día del año, y se supone una vida útil, sin actualizar ningún componente, de 4 años.
- (6) Se supone que el mando se usa 15 minutos al día (0,25 horas) y tiene una vida útil de 4 años.

Costes indirectos

Por último, analizaremos los costes indirectos:

- Electricidad: el precio de la electricidad en España ronda los 0,12€/kWh [8], y el ordenador se va a usar 480 horas con una potencia de unos 390W (0,39kW) [9]. Así pues, tenemos un coste de $480 \cdot 0{,}39 \cdot 0{,}12 = 22{,}464$€.
- Internet: el precio medio mensual de una conexión a Internet de hasta 30Mbps (no es necesario más puesto que sólo es usada para buscar información y para comunicaciones con el director) es de 41,3€/mes [10]. Así pues, sabiendo que este proyecto abarca desde febrero hasta junio, tenemos 5 meses: $41{,}3 \cdot 5 = 206{,}5$€.

Presupuesto total

Así pues, tenemos un coste total del proyecto de 60.570 + 33.85 + 54.73 + 22.464 + 206.5 = 60.887,544€. No obstante, este presupuesto inicial puede variar a lo largo de la ejecución del proyecto debido a obstáculos o situaciones imprevistas. Con objeto de minimizar dichas desviaciones, y como bien se explicaba anteriormente, las horas de cada tarea se calcularon al alza, añadiendo siempre un extra de contingencia. Al estar el presupuesto calculado en función de las horas de trabajo de cada tarea, el coste económico aquí expuesto es, en principio, más elevado del que debería corresponder. Además, los sueldos de los involucrados en el proyecto también se ha estimado al alza, como una forma de añadir contingencia adicional.

Anexo C - Descripción y uso del código

A continuación, se presenta brevemente la estructura del código usado y la funcionalidad de las clases. Para una explicación más detallada del programa, los archivos fuente están comentados para su fácil comprensión.

Explicación de las clases

- **Main:** es la clase principal, usada como recoger los datos de entrenamiento, crear modelos, entrenarlos y probarlos con sencillas llamadas a otras clases que se encargan de la parte compleja.
- **Wrapper:** se trata de la clase encargada de unir los distintos componentes del dominio y hacerlas funcionar. Es la clase que es llamada por el main, y contiene funciones para recoger datos, crear modelos, entrenarlos y ponerlos a prueba.
- **DrivingModel:** es la clase que permite crear, cargar, guardar, entrenar y usar modelos de conducción automática. Se ha hecho de forma parametrizable, para que con simples llamadas externas se puedan especificar hiper-parámetros, rutas y demás.
- **SpeedModel:** esta clase es la que permite crear, cargar, guardar, entrenar y usar un modelo de reconocimiento de dígitos. Se usa para extraer la velocidad en pantalla y pasársela después al modelo de conducción correspondiente.
- **Controller:** es la que se encarga de recoger los datos del mando de Xbox One en todo momento durante el entrenamiento.
- **Monitoring:** cuando se realiza la puesta a prueba de un modelo, esta clase es la encargada de registrar los errores cuando el usuario se así se lo especifica, guardando la velocidad, el estado de los pedales y el volante, realizando una captura de pantalla tanto de lo que se ve realmente como de lo que percibe el modelo, y calculando a cada momento el MTBF.
- **Result processer:** una vez realizadas la prueba de un modelo, se visualizan los resultados y extraen datos numéricos como el MSE o la desviación típica con esta clase. También es la que permite explorar los datos de entrenamiento.
- **SpeedTest:** clase usada de forma interna para comprobar que el modelo de la velocidad funciona correctamente.

Nota: el código está en inglés por costumbre y porque permite una mayor comprensión por parte de estudiantes o investigadores internacionales en caso de que deseen continuar el proyecto o usar sus resultados.